Novel Word Recognition and Word Spotting Systems for Offline Urdu Handwriting

Malik Waqas Sagheer

A Thesis

In

The Department

of

Computer Science and Software Engineering

Presented in Partial Fulfillment of the Requirements

For the Degree of Master of Computer Science at

Concordia University

Montreal, Quebec, Canada

February 2010

Your file Votre référence
ISBN: 978-0-494-67229-7
Our file Notre référence
ISBN: 978-0-494-67229-7

ABSTRACT

Novel Word Recognition and Word Spotting Systems for Offline Urdu Handwriting

Malik Waqas Sagheer

Word recognition for offline Arabic, Farsi and Urdu handwriting is a subject which has attained much attention in the OCR field. This thesis presents the implementations of offline Urdu Handwritten Word Recognition (HWR) and an Urdu word spotting technique. This thesis first introduces the creation of several offline CENPARMI Urdu databases. These databases were necessary for offline Urdu HWR experiments. The holistic-based recognition approach was followed for the Urdu HWR system. In this system, the basic pre-processing of images was performed. In the feature extraction phase, the gradient and structural features were extracted from greyscale and binary word images, respectively. This recognition system extracted 592 feature sets and these features helped in improving the recognition results. The system was trained and tested on 57 words. Overall, we achieved a 97 % accuracy rate for handwritten word recognition by using the SVM classifier.

Our word spotting technique used the holistic HWR system for recognition purposes. This word spotting system consisted of two processes: the segmentation of handwritten connected components and diacritics from Urdu text lines and the word spotting algorithm. A small database of handwritten text pages was created for testing the word spotting system. This database consisted of texts from ten Urdu native speakers. The rule-based segmentation system was applied for segmentation (or extracting) for handwritten Urdu subwords or connected components from text lines. We achieved a 92% correct segmentation rate for 372 text lines.

In the word spotting algorithm, the candidate words were generated from the segmented connected components. These candidate words were sent to the holistic HWR system, which extracted the features and tried to recognize each image as one of the 57 words. After classification, each image was sent to the verification/rejection phase, which helped in rejecting the maximum number of unseen (raw data) images. Overall, we achieved a 50% word spotting precision at a 70% recall rate.

ACKNOWLEDGMENTS

I send my deepest gratitude to my supervisor, Dr. Ching Y. Suen, whose precious guidance, support and encouragement were pivotal in establishing my self-confidence in this endeavour. I would like to thank everyone in the CENPARMI lab group. They were extremely supportive and helpful. Overall, CENPARMI was the ideal place to share ideas, knowledge and to work on my research. I would like to give special thanks to the Research Manager of the group, Mr. Nicola Nobile. He was very supportive throughout my stay at the CENPARMI lab. He was also very helpful in creation of CENPARMI Urdu databases and in listening to my new ideas regarding this thesis. A special thanks to CENPARMI member, Ms. Chun Lei He, for sharing her wealth of knowledge and providing immeasurable support. Also, I am grateful to Ms. Shira Katz for her editing and proofreading of my thesis.

I would like to thank all my friends, especially Hasan, Yousuf, Faju, Imran, Zeenat, and Muddassir, for their prayers and their immense support during my studies. Finally, the greatest thanks goes to my family, for their unconditional love, especially my brother, Ammar, and my uncle, Dr. Tanveer Sajid, who made it possible for me to follow my dreams and achieve my goals.

To:

My Father, My Mother, My Sisters and My Aunt Bibi Gul who loves me as her son.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

Introduction

1.1 Introduction

Pattern Recognition and Image Processing are important and progressive subjects in the field of computer science. It is always very captivating to find ways of enabling computers to understand and perform just like humans, i.e., to read, understand and recognize images [67].

There are thousands of languages being spoken and written around the world. Handwriting has been developed over hundreds of years as a way of communication and also to expand the human memory. Nowadays, handwriting recognition has become an important and challenging subject within the pattern recognition community because of the difficulty in recognizing data such as cheques and postal addresses that are written by hand. Also, nowadays, some electronic devices like Personal Digital Assistants (PDAs) and special digitizers are giving options to people to communicate through handwriting. There has been much research on Optical Character Recognition (OCR) since 1950 to improve the computer's capability to understand handwriting [67].

Off-line recognition and on-line recognition are two different ways of recognizing handwriting. In off-line recognition, the handwritten data is scanned from the paper and converted into a digital format, and in on-line recognition, the handwritten data is extracted in real time by writing on special digitizer screens with a special pen. The recognition rate is higher for on-line recognition systems because they have fewer recognition problems for unconstrained written words as compared to off-line recognition systems [68]. Strokes, direction of strokes and dynamic information can lead to a good accuracy in on-line recognition. Still, off-line recognition is very useful in the recognition of addresses and cheques recognition, signature verification and in the recognition of historical handwritten documents.

There are two kinds of handwriting recognition, one is word recognition and other one is digit recognition. Word recognition is a little more complicated as there is a higher number of letters than the number of digits in any language. Word recognition of the Arabic script is more difficult than Latin scripts because of its cursive nature. The Urdu is a famous language in South Asia. Its alphabet was derived from the Persian alphabet, which itself has been derived from the Arabic alphabet. Like Arabic, Urdu is written from right to left. However, Urdu has more isolated letters (38) than Arabic (28) and Persian (32) as shown in Fig. 1.2. This fact makes Urdu different from Arabic and Persian in appearance in such a way that it uses a slightly more complicated and complex script. Some Urdu words are shown in Fig 1.1 below:

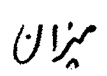
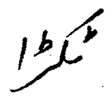
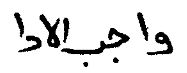
		
Balance	Article	Due

Fig. 1.1: Urdu Words

	URDU		FARSI		ARABIC	
1	Alif	ا	Alif	ا	Alif	ا
2	Be	ب	Be	ب	Be	ب
3	Pe	پ	Pe	پ	Te	ت
4	Te	ت	Te	ت	Se	ث
5	Tay	ٹ	Se	ث	Jim	ج
6	Se	ث	Jim	ج	Hey	ح
7	Jim	ج	Che	چ	Khe	خ
8	Che	چ	Hey	ح	Dal	د
9	Hey	ح	Khe	خ	Zal	ذ
10	Khe	خ	Dal	د	Rey	ر
11	Dal	د	Zal	ذ	Zey	ز
12	Daal	ڈ	Rey	ر	Seen	س
13	Zal	ذ	Zey	ز	Sheen	ش
14	Rey	ر	Zhe	ژ	Swad	ص
15	Arde	ڑ	Seen	س	Zvad	ض
16	Zey	ز	Sheen	ش	Toe	ط
17	Zhe	ژ	Swad	ص	Zoe	ظ
18	Seen	س	Zvad	ض	Ain	ع
19	Sheen	ش	Toe	ط	Gain	غ
20	Swad	ص	Zoe	ظ	Fey	ف
21	Zvad	ض	Ain	ع	Qaf	ق
22	Toe	ط	Gain	غ	Kaf	ك
23	Zoe	ظ	Fey	ف	Lam	ل
24	Ain	ع	Ghaf	ق	Mim	م
25	Gain	غ	Kaf	ک	Noon	ن
26	Fey	ف	Gaf	گ	Hey	ه
27	Qaf	ق	Lam	ل	Vao	و
28	Kaf	ک	Mim	م	Hay	ہ
29	Gaf	گ	Noon	ن	Yey	ي
30	Lam	ل	Vao	و		
31	Mim	م	Hay	ہ		
32	Noon	ن	Yey	ی		
33	Vao	و				
34	choti Hay	ہ				
35	He dow chashm	ھ				
36	Hamza	ء				
37	Choti Yey	ے				
38	Bardi Yey	ی				

Fig. 1.2: The Isolated Letters of Urdu, Farsi and Arabic

1.2 Research Overview

There has been very little research work done on online Urdu handwriting recognition [1-3] and almost none on offline Urdu handwriting recognition. In this thesis, we have created offline Urdu handwritten databases, worked on Urdu handwritten word recognition and on word spotting, for the very first time in the field of pattern recognition and image processing.

Our main focus of the research was to propose two systems for the Urdu language: the offline handwritten word recognition system and the word spotting system. For the creation of these systems, finding a good handwritten database was the main issue. However, there were no databases available for the Urdu language, so we had to start our research from creating comprehensive handwritten databases for the Urdu language [45]. The databases included dates, isolated digits, numeral strings, isolated characters, special symbols and 57 words of the Urdu Language. The data was collected with the help of data entry forms filled by native Urdu speakers. We also created the databases for handwritten text documents from ten different writers (also Urdu native speakers) at the Centre for Pattern Recognition and Machine Intelligence (CENPARMI) in Montreal. Each document consisted of four pages and each page consisted of different texts including those 57 words.

Our proposed word spotting system consists of different subsystems. The first and very important subsystem is Urdu offline handwritten word recognition. In this subsystem, we extracted the 464 global features from each image and trained our system on 57 words, taken from CENPARMI Urdu databases, where each dataset of one word represents one class. We used the Support Vector Machine (SVM) for classification purposes and we obtained the recognition accuracy of 96.02%. In our next subsystem, we worked on the segmentation of handwritten text lines which were taken from handwritten text document databases. Segmentation of words in Arabic languages is not easy and sometimes it is not possible because the inner-word space in a handwritten text line is bigger than the outer-word space. To overcome this problem, we proposed that instead of segmenting the words, we should segment the main connected components. In order to accomplish this goal, we performed the connected component analysis and Urdu diacritics recognition. Diacritics such as dots, double dots, etc., in Arabic scripts helped in the segmentation. In our proposed system, on the given text line, each diacritic was separated with its main component and each main component was segmented from the text line. In our final subsystem, we took the segmented connected components and applied our novel algorithm to spot the word. In this method, each word was recognized with the help of our first subsystem. Once the word was spotted, we performed the validation to make sure that we were not getting the garbage data. We used some rules (such as counting the number of dots in a word, height and width ratio of a word, etc.) for verification purposes and we also calculated the Euclidean Distance of each identified word from the mean of its class to make the rejection method strong.

1.3 Objectives of This Thesis

The main objectives of this thesis are:

- To propose the handwritten databases of the Urdu language. These standard databases could be useful for different research purposes such as Urdu numeral recognition, Urdu date recognition, Urdu character and word recognition and also Urdu word spotting.
- To propose the Handwritten Word recognition system by using global features. Therefore, there was no need for segmentation of a word for recognition purposes.

- To propose a word segmentation system using connected component analysis that could be useful for word spotting purposes.
- To propose and explore some challenges that could lead to a successful Urdu word spotting system.

This thesis is organized as follows: in Chapter 2, different approaches to handwritten word recognition and word spotting are discussed. Chapter 3 focuses on the creation of the CENPARMI Urdu databases. In Chapter 4, we discuss our handwritten word recognition system. Finally, Chapter 5 explains our text line segmentation and word spotting techniques.

CHAPTER 2

Literature Review: Approaches to Offline Handwritten Word Recognition and Word Spotting

2.1 Introduction

There are three different kinds of offline handwriting recognition: characters, words and numerals. In cursive languages, Handwritten Word Recognition (HWR) is more challenging and complex as compared to character and numeral recognition. For this reason, HWR for the cursive Arabic languages is the subject of significant research in the OCR field.

Some past research has been conducted on Urdu online handwriting recognition, but to date no significant work has been conducted on Urdu offline handwriting recognition. On the other hand, much research has been conducted on the recognition of Arabic and Farsi handwritten words, characters and numerals. In this chapter, because of the minimal work conducted on the Urdu language, we will focus on some state-of-the-art techniques for Arabic/Farsi HWR, word spotting and also some other techniques that are related to this research. This chapter is organized as follows: In section 2.2, different techniques for the implementation of Arabic/Farsi handwritten word recognition are discussed. In section 2.3, we discuss different methods that are used for Arabic text line segmentation and word spotting.

2.2 Handwritten Word Recognition

Many applications use offline HWR techniques, such as postal address reading for mail sorting purposes, cheque recognition and word spotting on a handwritten text page, etc.

The generalized offline HWR consists of five different components for recognition algorithms: pre-processing, representation, segmentation (only for segmentation-based recognition), feature extraction and classification [5]. The illustration of an HWR system structure is shown in Fig. 2.1, below:

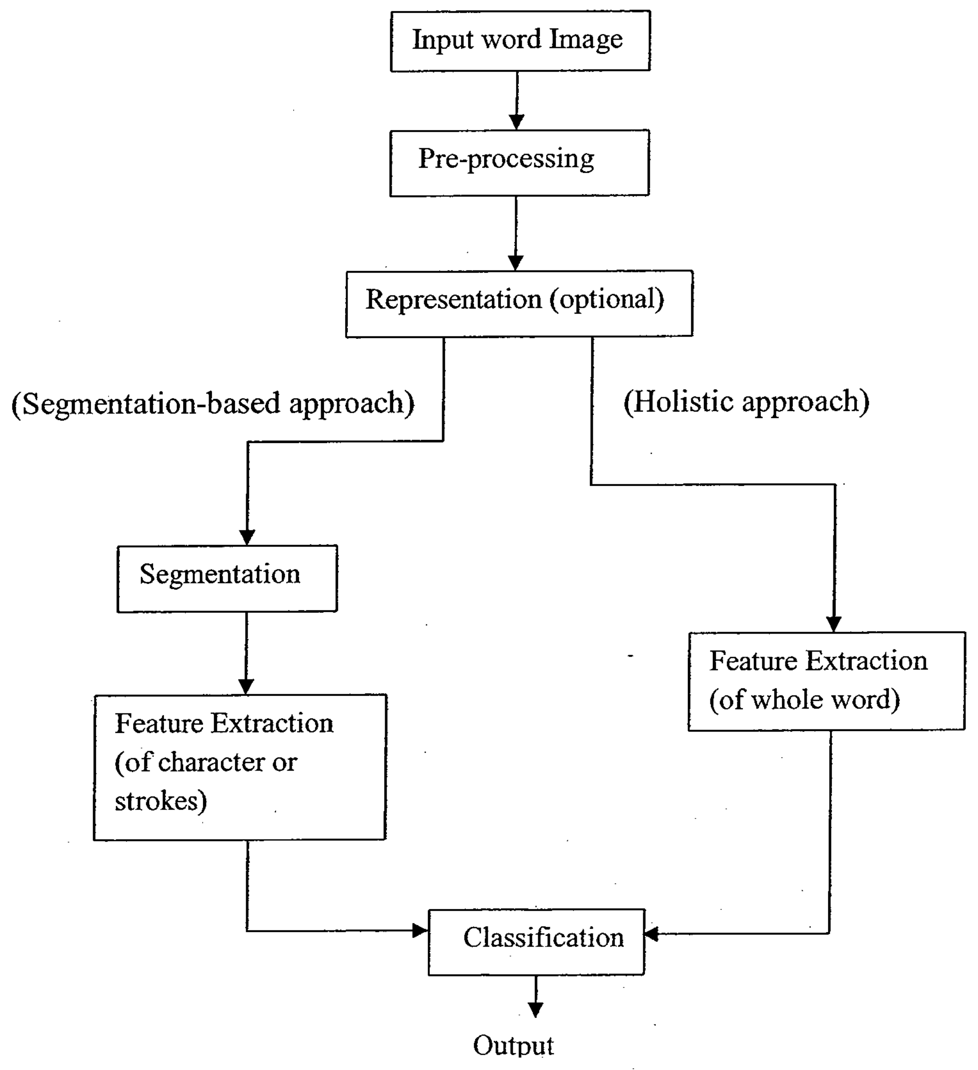

Input word Image
Pre-processing
Representation (optional)
(Segmentation-based approach)
(Holistic approach)
Segmentation
Feature Extraction (of character or strokes)
Feature Extraction (of whole word)
Classification
Output

Fig. 2.1: Generalized Structure for Handwritten Word Recognition System

In subsection 2.2.1, we will review the pre-processing phase. In subsection 2.2.2, we will discuss the representation phase. In subsection 2.2.3, we will review the different recognition approaches used in the recognition process.

2.2.1 Pre-processing

The first task for handwriting recognition is Pre-processing which is used for several different purposes mentioned below:

- To reduce or remove the noise in the text image
- To binarize the image
- To normalize the image
- To detect the baselines
- To correct the skew or slant

Pre-processing helps in improving the recognition results. It also helps in simplifying the feature extraction phase [6]. In Arabic HWR, two important pre-processing methods are baseline detection and slant correction. M. Dehghan et al. [9] and Rashaideh et al. [6] found the baseline in an Arabic text line by counting the maximum value in a horizontal projection histogram. They took the baseline of a skewed image by estimating the baseline that is rotated by an angle threshold between $+\theta$ and $-\theta$. Rashaideh et al.'s methods [10] are summarized in the following steps:

- They applied the horizontal projection and then calculated the maximum value.
- They rotated the image by angles of +20 degrees and -20 degrees and then applied the horizontal projection profiles and again calculated the maximum value.

- They compared the two peaks, and if the maximum value was higher than the previous maximum, they updated the maximum value, otherwise they stopped and determined the baseline. Detection of the baseline by using a horizontal projection is shown in Fig. 2.2, below:

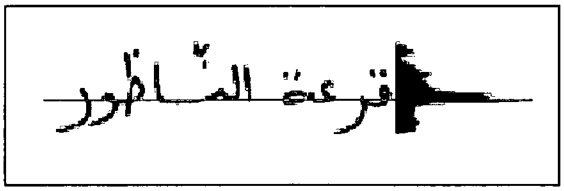


Fig. 2.2: Horizontal Projection Method for Detecting Baseline.

F. Farooq et al. [7] defined the importance of baseline detection for Arabic handwriting. Their research was based on the IFN/ENIT databases. They found the baseline using the local minima points of words. In Arabic text, the maximum value in a horizontal projection is usually the baseline. They were successful 78.5% of the time in locating the baseline. But in the case where the text had a large number of diacritics as compared to the main component, they were unsuccessful in finding the correct baseline. In another paper, J. AlKhateeb et al. [8] proposed that in the Arabic language, the baseline always lies below the middle line of the image. This method helped in improving the results. The results are shown in Fig. 2.3, below:

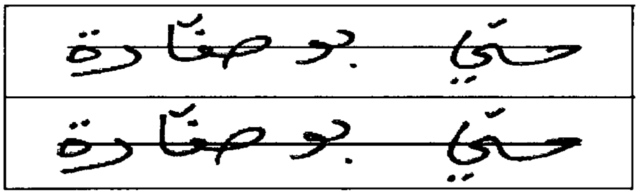


Fig. 2.3: Top Image shows failure in baseline detection by horizontal projection and bottom image shows the better outcome by using Alkhateeb et al.'s proposed algorithm [8] .

2.2.2 Representation

There are two well-known representations of images for HWR, Skeletons and Contours, and they are described below:

2.2.2.1 Skeletons

The skeleton image is the thinnest representation of an original image. Skeletons are usually one pixel wide. There are different algorithms ([6],[12],[14],[15]) that have been used to extract skeletons from an image. The skeleton of an Arabic word is shown in Fig. 2.4 (b), below:

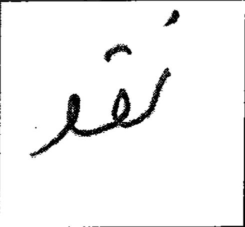
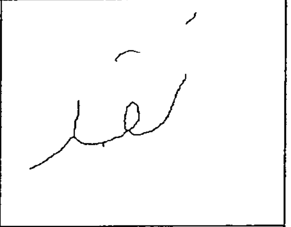
	
(a) Word	(b) Skeleton

Fig. 2.4: (a) Arabic Word and its (b) Skeleton

In 1996, Amin at al. [12] used skeletons of characters for their character recognition system. They extracted the following structural features from skeletons: End Points, Cross points, Branch points, number of Dots and their positions, Hamza (ء) and its position, loops, curves and lines. The End Points, Cross points and Branch points are shown in Fig. 2.5, below:

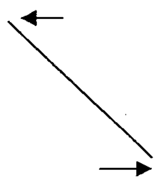
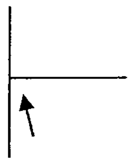
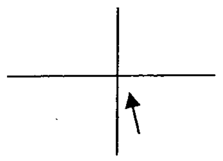
		
(a) End points	(b) Branch points	(c) Cross points

Fig. 2.5: Different Structural Feature Points

Amin at al. [12] used an artificial neural network which consisted of five layers. Two thousand (2000) characters were used to train the system and another thousand (1000) characters were used to test their recognition system. An accuracy rate of 92% was achieved. In another study, Abuhaiba et al. [13] used skeleton representation for their study on handwritten text recognition. The method adopted by them for feature extraction was the segmentation method, which is defined below:

(1) A word was segmented into subwords

(2) Subwords were segmented into strokes

(3) Strokes were segmented into small parts called tokens

The segmentation of a skeleton word into strokes and tokens is shown in Fig. 2.6, below:

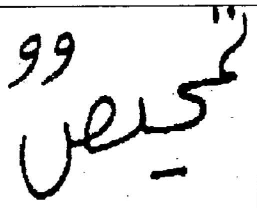
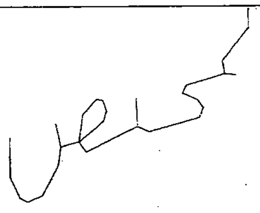
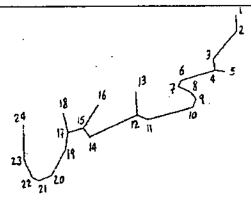
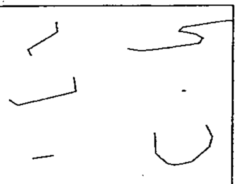
			
(a) Arabic word	(b) Skeleton	(c) Segmentation into Strokes	(d) Tokens

Fig. 2.6: Arabic word and skeleton of it's main part, and its segmentation into strokes and then into tokens[13].

In this study, each token was considered as a vertex that represented dots or loops or sequences of the vertex [16]. The "fuzzy sequential machine" was used as a recognizer of their system. Their system consisted of classes, sets of initial states and terminal states, stroke directions for entering states, and a function for transitioning between states. After the segmentation of strokes into tokens, each token passed through a recognition phase. If a token did not belong to any class, then that token was saved for the token learning process. A recognition rate of 55.4% was obtained for subwords, while a rate of 51.1% was achieved for character recognition.

2.2.2.2 Contours

Contours are defined as a set of border pixels or as the boundary of an image, and look like 2D polygons. Different algorithms ([17],[18],[19]) can be used to create contours of digital images. The contour of an Arabic word is shown in Fig. 2.7 (b), below:

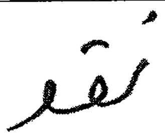
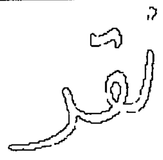
	
(a) Word	(b) Contour

Fig. 2.7: (a) Arabic Word, and (b) its contour

2.2.3 Recognition Approaches

There are two approaches used for the recognition phase in HWR: holistic and segmentation-based. In the holistic approach, each word is processed as a whole and there is no segmentation of a word into characters and in the segmentation-based approach, each word is first segmented into characters and each character is recognized independently [5]. In the next two subsections, we will review the state-of-art techniques for both approaches

2.2.3.1 Segmentation-Based Approaches

Arabic languages are written from right to left and characters connect with each other to make words. In Arabic languages, characters have different shapes depending on their position in the word. The same characters can have different shapes at starting, middle and ending points in a word as shown in Fig. 2.8, below:

	Character	Isolated	First	Middle	Last
1	Alef	(آ)ا	(آ)ا	ـا	ـا
2	Be	ب	بـ	ـبـ	ـب
3	Pe	پ	پـ	ـپـ	ـپ
4	Te	ت	تـ	ـتـ	ـت
5	Se	ث	ثـ	ـثـ	ـث
6	Jim	ج	جـ	ـجـ	ـج
7	Che	چ	چـ	ـچـ	ـچ
8	He	ح	حـ	ـحـ	ـح
9	Khe	خ	خـ	ـخـ	ـخ
10	Dal	د	د	ـد	ـد
11	Zal	ذ	ذ	ـذ	ـذ
12	Re	ر	ر	ـر	ـر
13	Ze	ز	ز	ـز	ـز
14	Zhe	ژ	ژ	ـژ	ـژ
15	Sin	س	سـ	ـسـ	ـس
16	Shin	ش	شـ	ـشـ	ـش
17	Sad	ص	صـ	ـصـ	ـص
18	Zad	ض	ضـ	ـضـ	ـض
19	Ta	ط	طـ	ـطـ	ـط
20	Za	ظ	ظـ	ـظـ	ـظ
21	Ayn	ع	عـ	ـعـ	ـع
22	Ghayn	غ	غـ	ـغـ	ـغ
23	Fe	ف	فـ	ـفـ	ـف
24	Ghaf	ق	قـ	ـقـ	ـق
25	Kaf	ک	کـ	ـکـ	ـک
26	Gaf	گ	گـ	ـگـ	ـگ
27	Lam	ل	لـ	ـلـ	ـل
28	Mim	م	مـ	ـمـ	ـم
29	Noon	ن	نـ	ـنـ	ـن
30	Waw	و	و	ـو	ـو
31	He	ه	هـ	ـهـ	ـه
32	Ye	ی	یـ	ـیـ	ـی

Fig. 2.8: Farsi Characters and their Shapes According to Start, Middle and End Points in a Word[21].

In the segmentation-based recognition approach, a given word is first segmented into characters (according to starting, middle or ending points) or strokes. There are two kinds of character segmentation: Explicit Character Segmentation and Implicit Character Segmentation. In explicit character segmentation, characters are segmented before the recognition process of a word. In implicit character segmentation, characters are segmented during the recognition process of a word.

In 1999, Mostafa and Darwish [20] presented an explicit segmentation approach for a handwritten text in Arabic. Their proposed approach was based on the study of the upper and lower contours of the word. They approached the study using baseline-independent algorithms to detect lines, to segment words into characters, and to extract dots and they used chain codes for their segmentation algorithms. The segmentation results achieved in this study are shown in Table 2.1, below:

Table 2.1: Results of Character Segmentation [20]

Number of Characters	**No. of Writers**	**Correct Segmentation Rate**
7,922	14	97.7%

In 1985, Almuallim et al. [16] defined the segmentation process of Arabic words. They also defined the characteristics of Arabic characters like the number of dots, loops, and the similarity of different characters in Arabic words. In the segmentation process, they proposed the algorithm to find the start and end point of any character or stroke from a word. They used skeletons to extract cross points, branch points and end points for segmentation purposes. For classification purposes, they divided the strokes into five different groups: dots, hamza, characters with loops, characters without loops (ending on a cross point or a branch point) and characters without loops (ending on a line end). The breakdown of an Arabic word into characters is shown in Fig. 2.9, below:

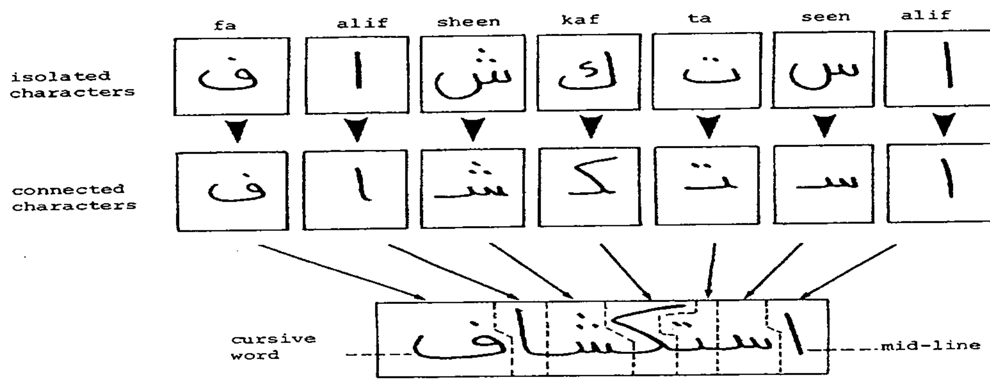


Fig. 2.9: Breakdown of an Arabic Word into Characters [16]

For feature extraction, they calculated the angle of the first part of the stroke curve from the starting point and another angle of the ending part of the stroke curve, as well as the height and width of the loop, if any. They trained their system on 400 words written by two individuals and achieved a rate of 81.25 % in the recognition results.

In another study (2005), Safabakhsh and Adibi [21] worked on the recognition of handwritten Farsi words in the Nastaaligh style (Fig. 2.10) and used the Continuous-Density Variable-Duration Hidden Markov Model (CDVDHMM [22]) for recognition. The Nastaaligh style may contain overlapping between different strokes in a word, which could become a problem in detecting a baseline or in ordering a character in a word. In this study, they defined the new algorithm to remove the ascenders, descenders and dots during the preprocessing stage of an image and they also proposed a segmentation algorithm based on analyzing the upper word contour. They used chain codes for contours to find the local minima of the upper word contour and then performed the segmentation process. Also, another process was proposed for the detection and segmentation of overlapped strokes in the word. In the feature extraction phase, they extracted eight features including the Fourier descriptors, the height and width ratio, the total number of loops, the position of left and right connections and the pixel densities from each segmented character. In this study, four reasons were defined that made the Hidden Markov Model (HMM) [26] a suitable classifier for their recognition process:

- The Markov model can easily code the sequential information that is useful for the Arabic nature of writing (that is also sequential).
- In an HWR system, HMM tries to find the sequence of segmented characters as hidden states, where these sequences were trained as observations during the observation process.
- There is no vector quantization error in the continuous symbol probability distribution and the properties of Arabic or Farsi characters can be modeled by the mixture of a Gaussian distribution.
- The overlapping problem can be fixed by a variable duration of states in HMM.

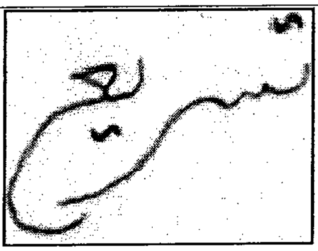
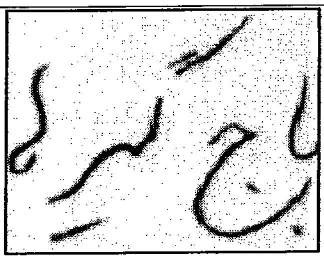
	
(a)	(b)

Fig. 2.10: Two Nastaaligh Style Handwritten Farsi Words [21]

For their experiments, they [21] used three data sets. The first dataset consisted of written words for training purposes. The second dataset consisted of 50 words written by seven different writers and they achieved a 69% recognition rate result on 5 iterations and a 91% recognition rate result on 20 iterations. They also tested their system on their third dataset which consisted of unknown words. For unknown words, their system achieved a 52.38% recognition rate on 5 iterations and a 90.48% recognition rate on 20 iterations. Overall, the segmentation-based recognition approach is more complex for the Arabic languages as compared to the holistic-based recognition approach. In the segmentation-based approach, the segmentation process should be very strong in order to achieve the highest recognition results.

2.2.3.2 Holistic Approaches

There is no segmentation of words required in the holistic approach and words are recognized as a whole. Different approaches to implement holistic-based handwritten Arabic / Farsi word recognition have been proposed using HMM ([19],[23-25]). For the recognition process, the HMM classifier has been widely used for handwritten as well as for printed Arabic scripts because of the cursive nature of Arabic / Farsi languages [23].

In 2005, Al-Hajj Mohamad et al. [23] proposed a one-dimensional HMM holistic-based offline handwriting recognition system for the Arabic language. In their proposed system, they extracted language-independent features as well as baseline-dependent features from binarized word images. Feature vectors were extracted by using a sliding window technique. They used the HMM classifier for recognition and IFN/ENIT [26] databases, which included Tunisian city names. An overview of their system is shown in Fig. 2.11, below:

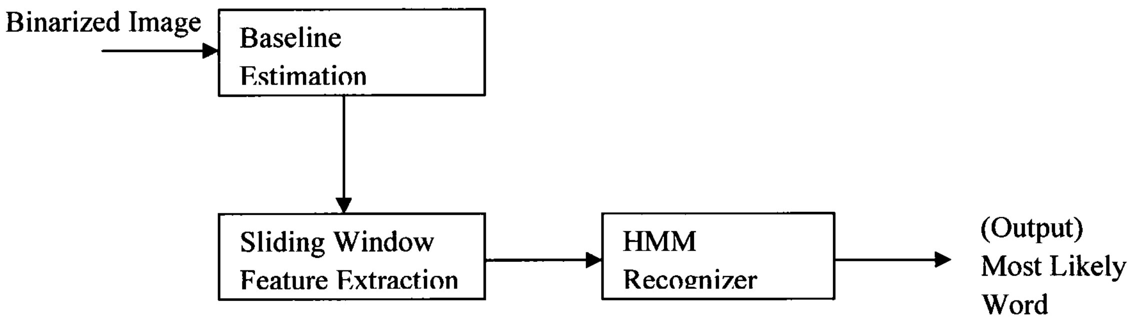


Fig. 2.11: Block Diagram of the Recognition System [23]

In the baseline estimation phase, they extracted the upper and lower baselines from the word image and these baselines divided the word image into three different zones, as shown in Fig. 2.12. The middle zone did not contain any ascenders and descenders, but the other two zones could have contained ascenders and descenders. Their approach to find baselines was based on finding the maximum pixel density in a horizontal projection curve. The maximum of the projection profile of a word corresponded to the lower baseline and the maximum of the projection profile from the top to the lower baseline corresponded to the lower baseline.

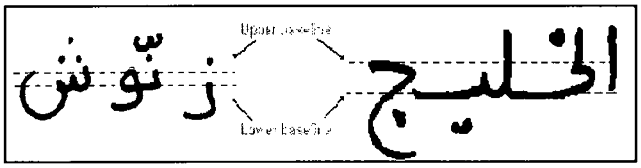


Fig. 2.12: Lower and Upper Baselines of Two Arabic Words [23]

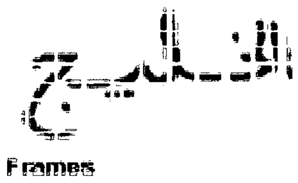


Fig. 2.13: Arabic word divided into frames

In the feature extraction phase, they divided the image into frames, as shown in Fig. 2.13, where each frame was divided into cells. The height of the frame was varied according to the size of the image but the size of the cell was fixed at 4 pixels. From each frame, they extracted a set of 24 features. Two different types of features were extracted by using a sliding window: distribution features and concavity features. Sixteen of those 24 features were distribution features based on black pixel densities and eight were concavity features. Concavity features provided the local concavity information and stroke information in each frame, as shown in Fig 2.14, below:

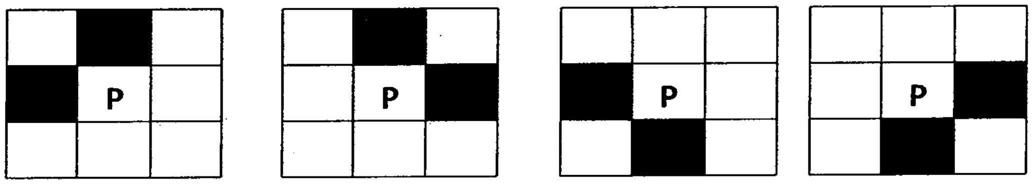


Fig. 2.14: Four kinds of concavity features for a pixel P [23]

In this study, they used the right-left HMM model (Fig. 2.15) for recognition. Also, a character model of HMM was used to implement the word model, where they found the following characteristics:

- Four states in the model
- Three transitions in each state
- Three Gaussian distribution mixtures used for each state

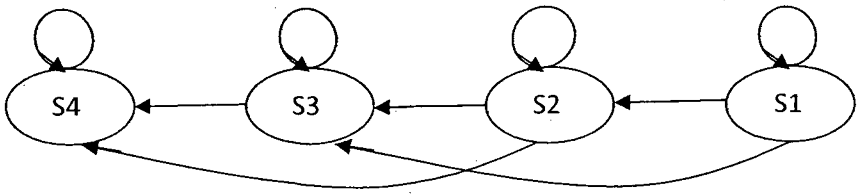


Fig. 2.15: A four-state Right-Left HMM model, where S Represents States.

They trained their system by using character-HMM parameters on whole words instead of on a character by character basis. With this approach, they first tested their system on 15 features without baseline detection and achieved a 74.90% recognition rate result. Then, when they detected baselines, their recognition results improved to 86.51%. So, baseline-dependent features helped in improving the recognition result by 11.61%.

In another study, M. Dehghan et al. [19] proposed a holistic approach to recognize handwritten Farsi words, by using a discrete HMM. Their lexicon size consisted of more the 17,000 images of 198 city names and they used greyscale images with 300 dpi (Dots Per Inch) resolution. They used the contour representation of images and saved the chain codes of each pixel from the contours. In the feature extraction phase, the image was divided into frames of a fixed size and there was a 50% overlap between two successive frames. The width of the frames was estimated to be twice the average stroke width that was calculated in the pre-processing phase.

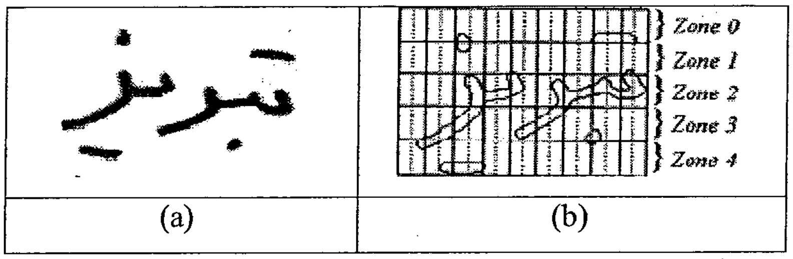


Fig. 2.16: (a) Arabic word, and (b) Contour of the Word in (a) and Division of the Image into Zones [19]

Each frame was divided into five zones of equal heights, as shown in Fig. 2.16. From each zone, they calculated a local histogram of chain codes. They used the right-left

HMM model for recognition, where they achieved a 32.04% recognition rate for their top one choice and a 93.63% recognition rate for their top 20 choices.

2.3 Handwritten Word spotting

In the word spotting or keyword matching technique, a handwritten text document image is scanned and then the word spotting program tries to locate a query word on that text document image. If the query word exists on the page, then the program will look for the coordinates of that word on the image. For Arabic languages, word spotting is not an easy task because sometimes the inner-word space of a word is larger than the outer-word space. Also, there is often overlapping of different words or within a word. There are two ways of spotting the words, segmentation-based and non segmentation-based. In segmentation-based word spotting, a given text page is segmented into text lines and then text lines are segmented into words. In non segmentation-based word spotting, a given query word is spotted directly on the page without any segmentation of text lines into words.

There have been several studies on Arabic/Farsi word spotting ([27-30]). In 2008, Raid Saabani and Jihad El-Sana [30] proposed a non segmentation-based (word) Arabic handwritten word spotting algorithm. Before applying the spotting algorithm, baselines of the rows of the given image were aligned horizontally, followed by segmentation of a text page into text lines and each connected component on a text line was labelled. For image representation, contours of connected components were used. In this study, they divided the connected components into two categories, main component and secondary component. Diacritics such as Dots and Hamzas were called secondary components. They also defined subwords such that both the main component and its secondary

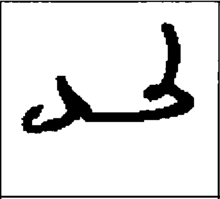
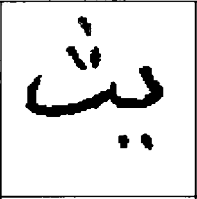
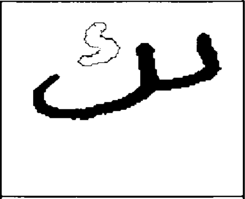
		
(a)	(b)	(c)

Fig. 2.17: Connected Components (Main Component (in black) and Secondary Component (in red and blue)) of Arabic Words [30]

Fig. 2.17 (a) shows a word and there is no secondary component involved and the other two images (b and c) show two subwords consisting of main component and secondary components.

An algorithm was used to simplify the contour image by converting it into a vertices format. In the feature extraction phase, they extracted geometric features (which are usually used for online Arabic handwriting recognition). They extracted vertices from the contour, and from the vertices they extracted the following features:

- The angle between two vectors
- The length of a vector
- The total number of loops in a connected component
- The length of a contour

In this study [30], two classifiers, Dynamic Time Warping (DTW) [31] and HMM, were used for matching two similar images. However, those images which were really different were not used for matching. They calculated horizontal and vertical histograms from the contours of two matching images. They used a threshold value for rejecting unknown images. They also counted the number of diacritics and their positions in the image for rejection or verification of a test image and called it a rule-based system.

Similarity in two images with the help of horizontal and vertical histograms is shown in Fig. 2.18, below:

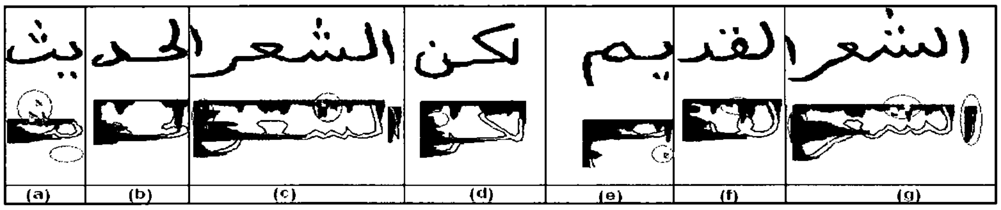


Fig. 2.18: Out of Seven Images, Two Images (c) and (g) show similarity in vertical and horizontal histograms [30]

Forty Arabic handwritten pages which consisted of more than 8000 words and 15,000 subwords were used. They achieved 82% and 70% recognition rate results using DTW and HMM, respectively, for the small training set. They achieved 86% and 76% recognition rate results for the large training set. They found that DTW gives a better performance than HMM.

In another study, S. N. Srihari et al. [28], in 2005, proposed an algorithm for handwritten word spotting on a text page by using a software called CEDARABIC. The CEDARABIC system was developed for spotting an Arabic word (it inherited most of its functionalities from the CEDAR-FOX software, developed by U.S. law enforcement for forensic examinations [32]). In this study, they focused on the following:

- The data collection for designing and testing of a system
- The user Interface of CEDARABIC
- Word segmentation
- Word-shape matching algorithm

Ten writers contributed in their data collection process and each of them wrote on 10 pages in the Arabic language. The databases consisted of 20,000 word images. In the word segmentation process, each line was divided into a set of Connected Components (CC). Contour representation was used for each connected component. Connected components were divided into two categories, major components and minor components or diacritics. Diacritics were found above and below the major components and were merged with major components to make clusters.

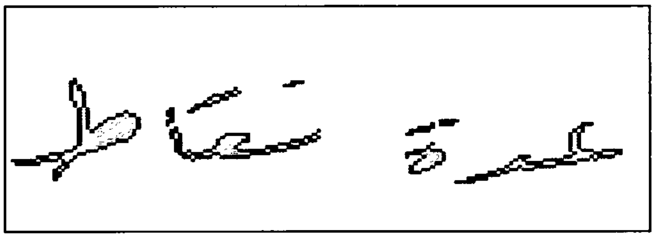


Fig. 2.19: Contours of Connected Components [28]

In this study, they found that the Alif character (ا) was a strong indicator for finding a word gap between two clusters. The height and width of a component helped in finding the Alif character. They used the following nine features to perform the segmentation by applying a neural network [33]: (1) The width of the first cluster, (2) The width of the second cluster, (3) The difference between the bounding boxes of two clusters, (4) The presence of the Alif character in the first cluster, (5) The presence of the Alif character in the second cluster, (6) The total number of components in the first cluster, (7) The total number of components in the second cluster, (8) The minimum distance between convex hulls around the two clusters, and (9) The ratio of the sum of the area of each cluster to the total area of both clusters.

The distance between convex hulls of the two clusters in an Arabic word is shown in Fig. 2.20, below:

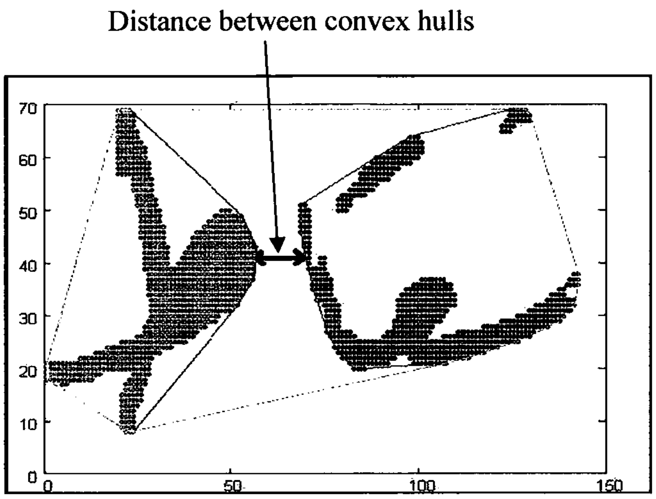


Fig. 2.20: Gaps Between Two Clusters and Convex Hull Around One Cluster in the Arabic Language are Shown by the Red Line and gaps around Both Clusters are shown by the Green Line

In their spotting method, their system used the global word shape features to spot the word. They also used 1024 binary features for matching the words with the help of a correlation similarity measurement that is given below:

$$d(X,Y) = \frac{1}{2}\left(1 - \frac{s_{11}s_{00} - s_{10}s_{01}}{[(s_{10}+s_{11})(s_{01}+s_{00})(s_{11}+s_{01})(s_{00}+s_{10})]^{\frac{1}{2}}}\right) \tag{2.1}$$

Where, X and Y are two feature vectors and s_{ij} represents i'th and j'th bit values in X and Y [28].

Y [28].

Overall, in this study, a 60% correct segmentation rate was achieved and an accuracy rate of 70% was achieved for word spotting, when their system was trained on the writing of eight writers.

In this chapter, some proposed approaches for offline Arabic / Farsi word recognition and word spotting were reviewed. In the next chapter, we will discuss our Urdu database creation.

CHAPTER 3

CENPARMI Off-Line Handwritten Urdu Databases

3.1 Introduction

Urdu is an Indo-European [34] language originating in India. It is a popular language in South Asia. Urdu is one of the 23 official languages in India and it is one of the two official languages in Pakistan. Urdu is also one of the most spoken languages around the world [46].

Written Urdu was derived from the Persian alphabet, which itself was derived from the Arabic alphabet. Urdu uses almost 70% of the grammar from the Persian language, and the rest comes from Arabic and Turkic languages. Muslims started using this language in the 1600's in India. After the 14th of August, 1947, when Pakistan was separated from India, Urdu became the national language of Pakistan and today more than 200 million people use it in the Indian subcontinent.

Finding a good database is a common issue in handwriting recognition. There is only one known offline handwritten Urdu database that was created at the Centre for Pattern Recognition and Machine Intelligence (CENPARMI) [45]. This Urdu database can be used for multiple applications, and it will be used for our experiments. The value of this Urdu database is based on the variety of its contents. The database contains a large number of Urdu isolated digits and numeral strings of various lengths, including those with decimal points. This database can be used for various applications such as digit, letter, word, and cheque recognition as well as for word spotting.

This chapter is divided into seven sections. In Section 3.2, we describe the related work in HWR. In Section 3.3, we describe the data collection process. Section 3.4 describes the data extraction and includes a description of pre-processing methods. In Section 3.5, we discuss the summary of the database followed by descriptions of the datasets. In Section 3.6, we explain the ground truth data. In Section 3.7, we discuss the use of Urdu isolated numerals in previous recognition experiments. In Section 3.8, we discuss the conclusion.

3.2 Related Work

Different handwritten databases have been created for cursive languages, such as Arabic ([37-38],[42]), Farsi ([40],[43-44]), Pashto [41] and Dari [39], for different off-line handwritten recognition purposes.

In 2002, an offline database called IFN/ENIT was created for the Institute for Communications Technology (IFN) by the Technical University Braunschweig in Germany and Ecole Nationale d'Ingénieur de Tunis (ENIT) in Tunisia. There were 411 writers who contributed to creating this database and it consisted of more than 26,400 images of 937 town/village names. This database also provided the ground truth information of each image such as the sequence of character shapes and the baseline information. The IFN/ENIT database was divided into four different sets for testing and training purposes. In 2008 [43], a similar work was done for the creation of a database in Farsi handwriting.

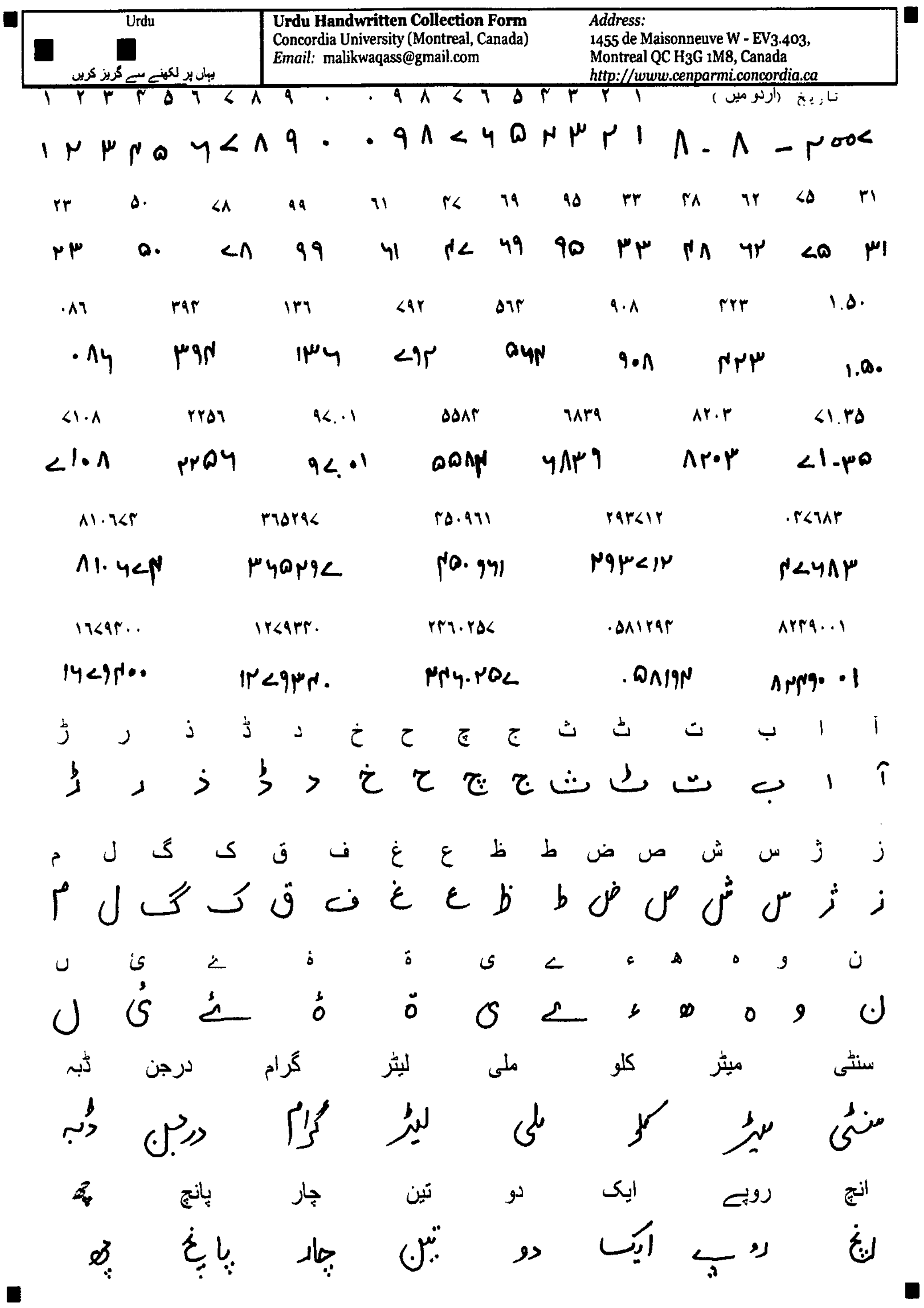
In 2008, the Centre for Pattern Recognition and Machine Intelligence (CENPARMI) at Concordia University in Canada had researchers working for the first time in the creation of handwritten databases for the Dari [39] and Arabic languages [38]. These databases consisted of handwritten dates, isolated digits, numeral strings, isolated characters and words and special symbols used in each language. About 200 writers contributed to the creation of the Dari databases and 328 writers contributed to the Arabic databases. Special forms were designed and used for the collection of data. Both databases were created in binary and greyscale formats. These databases also provided the ground truth information of each image such as the writer's personal information and the contents of each image. The databases were divided into three sets for training, testing and validating purposes.

At CENPARMI, researchers collected data for Dari, Pashto [41], Farsi [40], Arabic, and Urdu languages all at the same time. We followed the same method as other researchers for the creation of each database. Therefore our database structure is similar to CENPARMI's Dari and Arabic databases.

3.3 Data Collection Process

Much effort was spent on the collection of Urdu handwritten data. To start things off, a two-page data entry form was designed for the collection of handwritten data. We started the collection process in Montreal and Winnipeg, in Canada, but the data collection was limited so we moved our efforts to Pakistan, where 70% of the data was collected.

The first page of the form contains 20 Indian isolated digits (two samples of each digit), a freestyle handwritten date, 38 numeral strings of various lengths, 37 isolated characters, 6 special isolated characters and 16 words. The second page includes the remaining 41 words, five special symbols and the writer's personal information (at the bottom of the form). Fig. 3.1 and 3.2 show samples of a filled form.



Urdu

یہاں پر لکھنے سے گریز کریں

Urdu Handwritten Collection Form
Concordia University (Montreal, Canada)
Email: malikwaqass@gmail.com

Address:
1455 de Maisonneuve W - EV3.403,
Montreal QC H3G 1M8, Canada
http://www.cenparmi.concordia.ca

تاریخ (اردو میں) ۱ ۲ ۳ ۴ ۵ ۶ ۷ ۸ ۹ ۰ ۰ ۹ ۸ ۷ ۶ ۵ ۴ ۳ ۲ ۱

۸ - ۸ - ۲۰۰۷ ۱ ۲ ۳ ۴ ۵ ۶ ۷ ۸ ۹ ۰ ۰ ۹ ۸ ۷ ۶ ۵ ۴ ۳ ۲ ۱

۳۱ ۷۵ ۶۲ ۴۸ ۳۳ ۹۵ ۶۹ ۴۷ ۶۱ ۹۹ ۷۸ ۵۰ ۲۳

۳۱ ۷۵ ۶۲ ۴۸ ۳۳ ۹۵ ۶۹ ۴۷ ۶۱ ۹۹ ۷۸ ۵۰ ۲۳

۱.۵۰ ۴۲۳ ۹۰۸ ۵۶۴ ۷۹۲ ۱۳۶ ۳۹۴ ۰۸۶

۱.۵۰ ۴۲۳ ۹۰۸ ۵۶۴ ۷۹۲ ۱۳۶ ۳۹۴ ۰۸۶

۷۱.۳۵ ۸۲۰۳ ۶۸۳۹ ۵۵۸۴ ۹۷.۰۱ ۲۲۵۶ ۷۱۰۸

۷۱.۳۵ ۸۲۰۳ ۶۸۳۹ ۵۵۸۴ ۹۷.۰۱ ۲۲۵۶ ۷۱۰۸

۰۴۷۶۸۳ ۲۹۴۷۱۲ ۲۵۰۹۶۱ ۳۶۵۲۹۷ ۸۱۰۶۷۴

۴۷۶۸۳ ۲۹۴۷۱۲ ۲۵۰۹۶۱ ۳۶۵۲۹۷ ۸۱۰۶۷۴

۸۲۳۹۰۰۱ ۰۵۸۱۲۹۴ ۲۳۶۰۲۵۷ ۱۲۷۹۳۳۰ ۱۶۷۹۴۰۰

۸۲۳۹۰۰۱ ۰۵۸۱۲۹۴ ۲۳۶۰۲۵۷ ۱۲۷۹۳۳۰ ۱۶۷۹۴۰۰

آ ا ب ت ٹ ث ج چ ح خ د ڈ ذ ر ڑ

آ ا ب ت ٹ ث ج چ ح خ د ڈ ذ ر ڑ

ز ژ س ش ص ض ط ظ ع غ ف ق ک گ ل م

ز ژ س ش ص ض ط ظ ع غ ف ق ک گ ل م

ن و ہ ھ ء ے ی ۃ ۂ ۓ ئ ں

ن و ہ ھ ء ے ی ۃ ۂ ۓ ئ ں

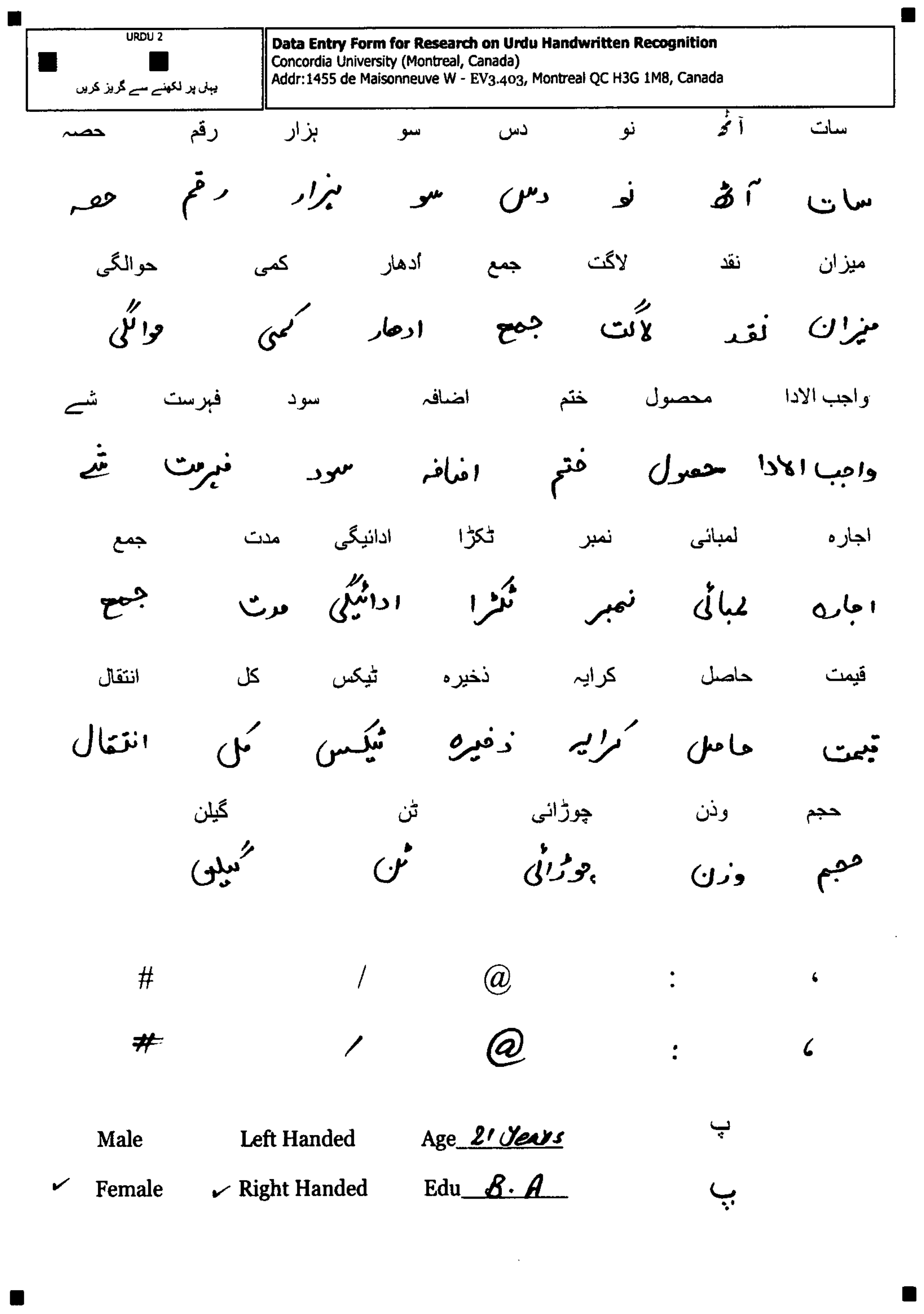
سنٹی میٹر کلو ملی لیٹر گرام درجن ڈبہ

سنٹی میٹر کلو ملی لیٹر گرام درجن ڈبہ

انچ روپے ایک دو تین چار پانچ چھ

انچ روپے ایک دو تین چار پانچ چھ

Fig. 3.1: Sample of a Filled Form (Page 1).



URDU 2

یہاں پر لکھنے سے گریز کریں

Data Entry Form for Research on Urdu Handwritten Recognition
Concordia University (Montreal, Canada)
Addr:1455 de Maisonneuve W - EV3.403, Montreal QC H3G 1M8, Canada

حصہ	رقم	ہزار	سو	دس	نو	آٹھ	سات
حصہ	رقم	ہزار	سو	دس	نو	آٹھ	سات
حوالگی	کمی	ادھار	جمع	لاگت	نقد		میزان
والگی	کمی	ادھار	جمع	لاگت	نقد		میزان
شے	فہرست	سود	اضافہ	ختم	محصول		واجب الادا
شے	فہرست	سود	اضافہ	ختم	محصول		واجب الادا
جمع	مدت	ادائیگی	ٹکڑا	نمبر	لمبائی		اجارہ
جمع	مدت	ادائیگی	ٹکڑا	نمبر	لمبائی		اجارہ
انتقال	کل	ٹیکس	ذخیرہ	کرایہ	حاصل		قیمت
انتقال	کل	ٹیکس	ذخیرہ	کرایہ	حاصل		قیمت
	گیلن	ٹن	چوڑائی	وزن	حجم		
	گیلن	ٹن	چوڑائی	وزن	حجم		

#	/	@	:	،
#	/	@	:	،

Male | Left Handed | Age 21 Years | پ

✓ Female | ✓ Right Handed | Edu B. A | پ

Fig. 3.2: Sample of a Filled Form (Page 2).

We gathered handwriting samples from 343 different writers. We kept track of the writer's sex, age, and handedness (left or right), as seen in the last part of Fig. 3.2. Even though for this research, the writer's personal information has no significance, it may be useful for future research. The writers were distributed into three categories as follows: right-handed males (75.4%), left-handed males (5.6%), and right-handed females (19.0%) [45]. We noticed that right-handed writers outnumbered the left-handed writers in our database. There were no left-handed female participants in our study.

3.4 Data Extraction and Pre-processing

Digital versions of all forms were obtained after scanning them. We saved those images as color (24 bit) TIFF images with a resolution of 300 Dots per Inch (DPI). The data extraction process was started by removing the red lines from the forms. Some writers exceeded the boundaries while writing on the forms, as shown in Fig.3.3. A special algorithm was used to remove the red lines from the digital images [45], as shown in Fig. 3.4. Removal of these red lines helped in extraction of handwritten data from the form.

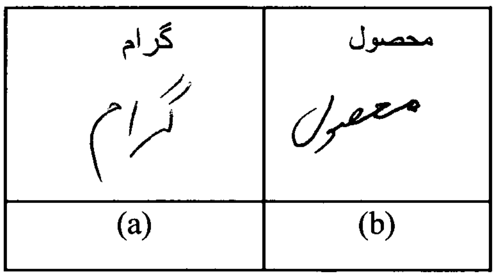


Fig. 3.3: Two Samples of Urdu Handwritten Words Exceeding the Field Boundaries.

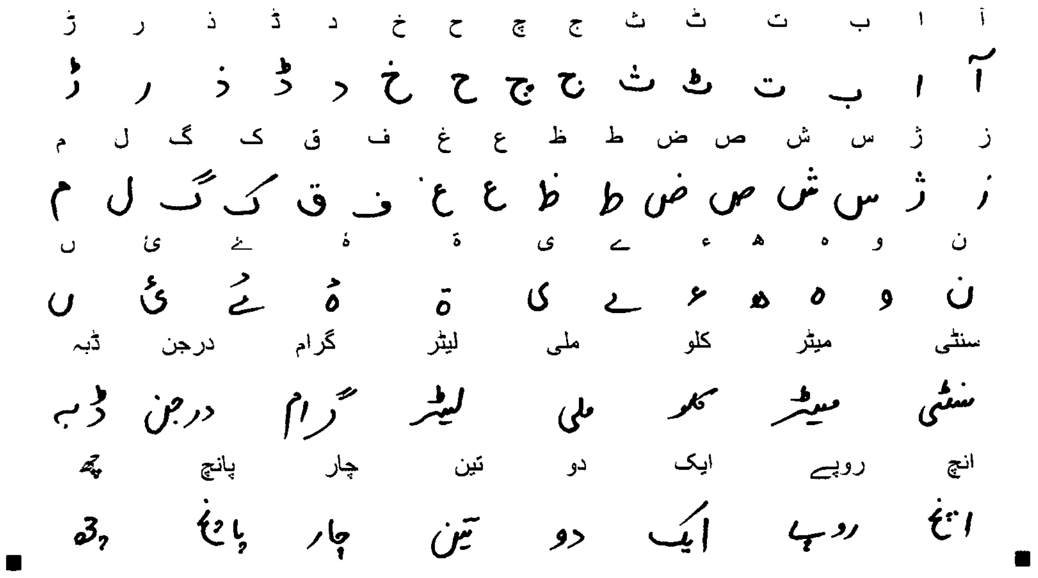

آ ا ب ت ٹ ث ج چ ح خ د ڈ ذ ر ڑ
ز ژ س ش ص ض ط ظ ع غ ف ق ک گ ل م
ن و ہ ھ ء ے ی ۃ ۂ ۓ ئ ں
سنٹی میٹر کلو ملی لیٹر گرام درجن ڈبہ
انچ روپے ایک دو تین چار پانچ چھ

Fig. 3.4: A Small Portion of a Form After Removing the Red Lines.

After removing the red lines, the box's coordinates were located for each handwritten sample. A special filter was applied to remove the salt and pepper noise on each image. There was also a great effort involved in manually cleaning those images. Besides true color, all the forms were saved in two other formats: greyscale and binary. Coordinates of the area for each handwritten element on the true color form were located manually. After the red lines were removed from the true color forms, a special program was developed to automate the data extraction process in the true color formats. By taking the coordinates of each box, this program extracted the box image in its true color. The images from the same box numbers of each form were saved in a unique folder. Once the databases were created in true color images they were then converted into greyscale and binary formats.

3.5 Database Overview

We created three databases in total: true color, greyscale, and binary formats. There were six basic datasets in each database: isolated digits, dates, isolated letters, numeral strings, words, and special symbols. In each dataset, the data was divided into training, validation, and testing sets by picking elements with the approximate distributions of 60%, 20%, and 20%, respectively.

The overall organization of the databases is summarized in the following diagram:

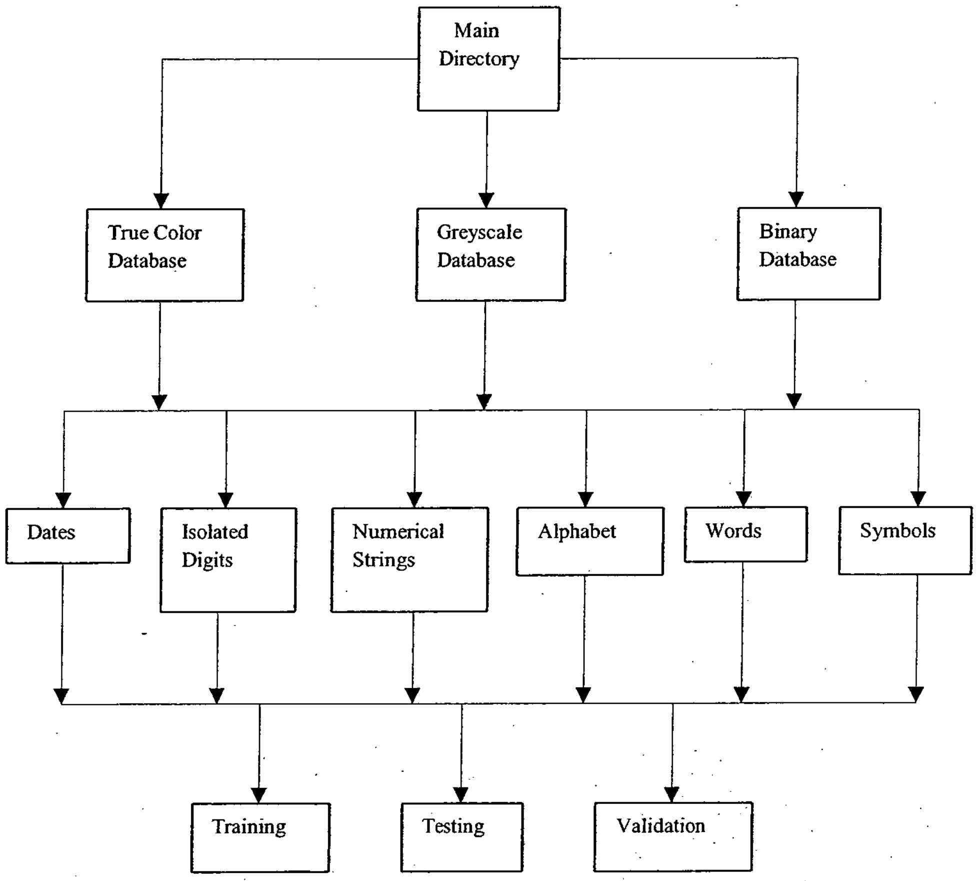


Fig. 3.5: The overall structure of Urdu Databases

A complete description and statistics of each dataset are given in the following subsections.

3.5.1 Urdu Date Dataset

In the Urdu language, there are multiple ways to write dates and the Gregorian calendar is followed for doing so. Writers were given the freedom to write the dates in any format. They sometimes used slash '/', hyphen '-'and/or dot '.' to separate the day, month and year. Some people drew a curvy line beneath the year (؁), which represents سن (san) in Urdu which means year in English, and some used the hamza 'ء' at the end of a year, which represents عیسوی (Aiswee) in Urdu which means A.D. in the Gregorian calendar. Some people also wrote the name of a month in Urdu letters (جولائی means July) instead of writing it in numeral format.

From the English influence, some people also used the dot to separate the month, day and year. This was a challenge in date recognition, since the dot is similar in shape to the Urdu digit zero. Some samples of Urdu dates are shown in the Table 3.1. Of the 318 date samples, 190 belonged to the training set and 64 images belonged to each of the validation and test sets.

Table 3.1: Different Samples of Urdu Dates.

English Date	Urdu Date
12/02/2007	۱۲/۰۲/۲۰۰۷
7-7-2007	۷-۷-۲۰۰۷
7 July, 2007	۷ جولائی ۲۰۰۷
16-Jul-2007 A.D	۱۶ جولائی ۲۰۰۷ء
8.8.2007	۸.۸.۲۰۰۷

3.5.2 Urdu Isolated Digit Dataset

Each form contained two samples of each isolated digit. In the numeral strings, each digit was repeated 14 to 18 times at different positions. We performed segmentation of the numeral strings by using a segmentation algorithm to separate and extract the digits. Segmentation was performed on the greyscale and binary images. After the segmentation, a set of isolated digits was created.

All of the other datasets are available in true color except for the isolated digits. We extracted a total of 60,329 digits, where 33,974 were selected for training, 13,177 for validation, and 13,178 for testing. A sample of each extracted digit is shown in Table 3.2.

Table 3.2: Samples of Handwritten Urdu Isolated Digits

Nine	Eight	Seven	Six	Five	Four	Three	Two	One	Zero
۹	۸	۷	۶	۵	۴	۳	۲	۱	۰

The Urdu isolated digits (dataset) from these databases were used for recognition experiments at CENPARMI [45]. Basic pre-processing was performed and 400 gradient features were extracted from each greyscale image. In this study, a very high accuracy of 98.61% was achieved on a test set by using the Support Vector Machine (SVM) classifier.

3.5.3 Urdu Numerical Strings Dataset

The data entry forms contained fields for 38 different numeral strings with varying lengths of 2,3,4,6 and 7 digits. Every digit, including the decimal point, was represented at least once in the strings (the details were explained in section 3.5.2). There are two ways in Urdu to write a decimal point; one way is to use a dot (.) and the second way is to use a hamza (ء). In most of the samples, the positions of the dot and the hamza in the numeral strings were located below the baseline. This could be a challenge in the recognition of real strings as the dot (.) looks like a zero in Urdu. We divided this dataset into two sets – an integer set and a real set. The integer set contained the numerical strings of lengths 2, 3, 4, 6 and 7. The real set included decimal numbers of lengths 3 and 4. Samples for integer and real numeral strings are shown in Table 3.3. A total of 12,914 strings were extracted, which were divided into training (7,750 strings), validation (2,584 strings), and testing (2,580 strings) sets.

Table 3.3: Samples of Different Numeral Strings

Type of String	English (Numeral String)	Urdu (Numeral string)
Integer	47	۴۷
Integer	2460257	۲۴۶۰۲۵۷
Real	1.50	۱٫۵۰
Real	1.50	۱.۵۰

3.5.4 Urdu Alphabet Dataset

The Urdu alphabet consists of 38 basic letters and five special characters. A special character may exist by itself or may consist of a combination of a letter and a diacritic. Each diacritic generates a different sound for the letter. We added some of these special characters to our data entry form, as shown in Table 3.4.

Similar to Arabic and Farsi [9], the words in Urdu are written in a cursive manner, and the shape of a letter in Urdu changes according to its position within a word. The data entry form included one sample of each of the 38 Urdu isolated letters and each of the special characters. There were 14,890 letters in total, including special characters, with 8,934 of the letters marked for training and 2,978 of the letters for each of the validation and testing sets.

Table 3.4: Some Special Urdu Characters

Alif Mud AA	Bardi ye Hamza	Noon Guna	Chooti Ye Hamza	Hey Hamza
آ	ئے	ں	ئ	ۂ

3.5.5 Urdu Words Dataset

We selected 57 Urdu (mostly finance-related) words for our word dataset. It included words for weights, measurements, and currencies. Samples of some of these Urdu words are shown in Table 3.5. The total number of extracted images was 19,432. These were divided into training (60%), validation (20%), and testing (20%) sets.

Table 3.5. Samples of Urdu Finance-Related Words

Cash	Thousand	Cost	Decrease	Balance	Amount
نقد	ہزار	لاگت	کمی	میزان	رقم

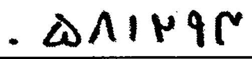
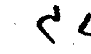
3.5.6 Urdu Special Symbols Dataset

The data entry form also included some special symbols that usually appear in any Urdu document. The following symbols, commonly used in other languages, were used: comma (,), colon (:), at (@), slash (/), and pound (#). The total number of training samples used was 1,020. The validation set and the testing set contained 340 and 345 images, respectively.

3.6 Ground Truth Data

For each folder that contained handwritten samples, we included the ground truth data file. For each sample, the ground truth data file included the following information: image name, content, number of connected components (CCs), writer number, length of a content, gender, and handwriting orientation. Also, the writer number, the box number and the page number of the data entry form could be found in the image name. An example of the ground truth data for the numeral strings dataset is shown in Table 3.6, below:

Table 3.6. Examples of the Ground Truth Data for Urdu Numeral Strings Dataset

Image Name	URD0110_P01_056.tif	URD0090_P01_029.tif
Content	0581294	47
Handwritten Content		
Writer No.	URD0110	URD0090
Page No. (Forms)	01	01
Box No.	56	29
Sex	F	M
Hand Orientation	R	R
Length (No. of CCs)	7	2

3.7 Conclusion:

This database could help in solving new and different challenges in the recognition of Urdu handwritten dates, isolated digits, strings, isolated characters, words, and symbols. We used the Urdu word dataset from this database for our Urdu handwritten word recognition system (holistic approach) and we will discuss this in detail in our next chapter. The Urdu word dataset will be applied to our word spotting system and is discussed in Chapter 5.

CHAPTER 4

Urdu Handwritten Word Recognition: A Holistic Approach

4.1 Introduction

Offline handwritten word recognition, especially cursive handwritten word recognition, has always been a challenging area in pattern recognition. Offline word recognition is usually complex as compared to offline numeral and character recognition. In this chapter, we will focus on our HWR system. As defined in chapter 2, an HWR system can be holistic-based or segmentation-based. Our HWR system is holistic-based and consists of the following phases:

- Pre-processing
- Feature extraction
- Recognition or Classification
- Experiments and results

In this chapter, we will define each phase in detail.

4.2 Pre-Processing

Pre-processing of images is an essential part of the off-line recognition phase. Recognition results rely on the pre-processing phase. A little noise on the image can result in incorrect recognition. Our pre-processing phase consists of the following steps:

1) During database creation, writers used either blue or black ink. Sometimes the color of the ink was very light or we lost the intensity of handwriting during our scanning process. In order to make the handwritten text consistent, we converted greyscale images into binary format based on a threshold value of 0.9.

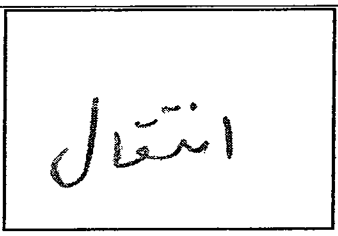
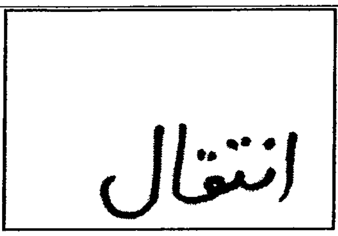
	
(a)	(b)

Fig. 4.1: Two Urdu Words with Different Ink Colors

Fig 4.1 shows two different color intensities in two different images. After converting them into binary format, both images had the same color intensities as shown in Fig. 4.2, below:

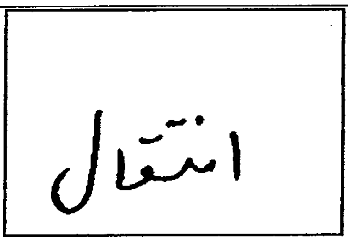
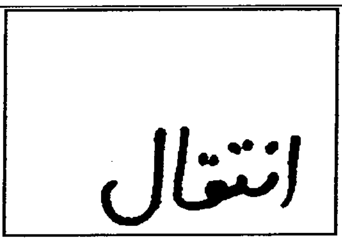
	
(a)	(b)

Fig. 4.2: Two Urdu Words with the Same Color Intensities.

2) Median filters were applied on the binary images to reduce the salt and pepper noise and to smooth the images. The size of each filter was 3-by-3 neighbourhood pixels. An example of noise removal from an Urdu word image is shown in Fig. 4.3, below:

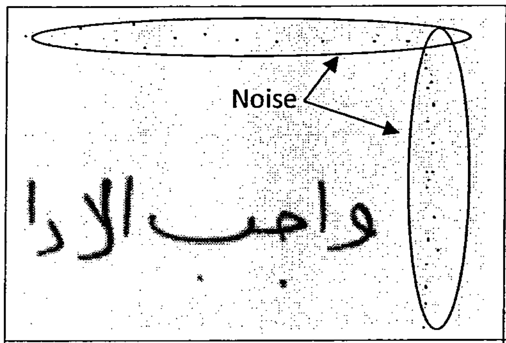
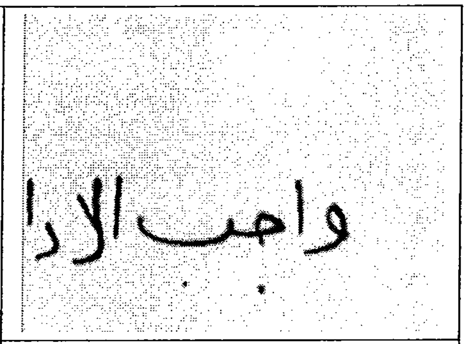
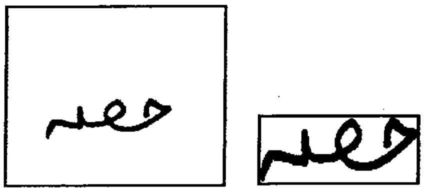
	
(a) Image with Noise	(b) Smooth Image after applying median filter

Fig. 4.3 : Image (a) Shows Some Noise and Image, and (b) Without Noise after Applying Median Filter

3) White space around the written word in an image does not help in any recognition process. So this unwanted white space around the word was eliminated. To eliminate this white space, bounding boxes were used. From each side of the binary image, the first pixel of the written word was located. This produced four points which formed the boundaries of the bounding box. The white area around this box could then be eliminated using these four values. The elimination of white space in an image is shown in Fig. 4.4, below:



(a) (b)

Fig. 4.4: Image (b) Shows Elimination of White Space from Image (a)

4) It is normal that everyone does not write with the same style and size. Before extracting the features, it was necessary to normalize the size of all images before the training and testing phases. Word size normalization is an important pre-

processing operation and each image was normalized to two different sizes, 64 x 64 pixels and 128 x 128 pixels (Fig. 4.5), by using an aspect ratio adaptive normalization strategy [47]. Two different sizes of the image were used for two different feature extraction processes, which will be explained in section 4.3.

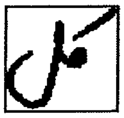
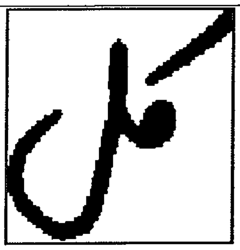
		
(a) Original Image	(b) 64x64	(c) 128x128

Fig. 4.5: Size Normalization of Urdu word (a) into different sizes (b) and (c)

The following figure (4.6) shows the overall pre-processing phase:

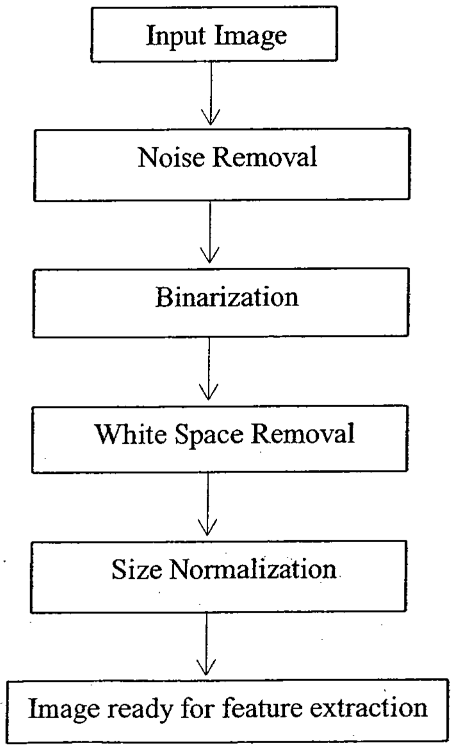


Fig. 4.6: Pre-processing Phase

4.3 Feature Extraction

Feature extraction is the most important phase of any recognition system. Locating relevant and good features will always lead to high recognition results. Different features are extracted depending on the problem's domain. For handwriting recognition, different features like chain codes, structural features, statistical features, curvature features, projection profile and gradient features (directional features) have been used in different studies ([4],[19],[36],[25],[48]).

In our study, we extracted two different types of features from each image: gradient features and structural features. We conducted various experiments with different features and found that the combination of these two features produced the best results for our HWR system. The Following sections will explain the extraction processes for both features.

4.3.1 Gradient Feature Extraction

Gradient features are directional features and can be extracted from the gradient of a greyscale image for a handwritten text. A gradient vector of discrete direction is decomposed into two components by specifying a number of standard directions (chaincode directions). The Decomposition of a gradient vector into two components is shown in Fig. 4.7, below:

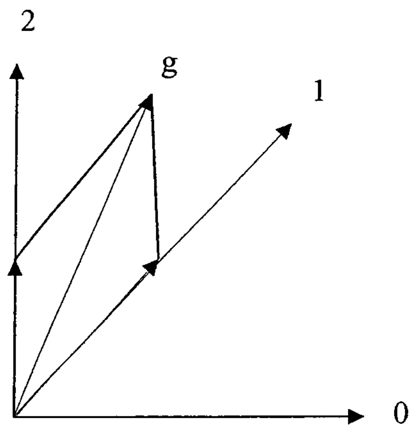


Fig. 4.7: Gradient Vector Decomposition, where g = Gradient Vector.

In 2002, M. Shi et al. [36] used gradient features and curvature features for handwritten numeral recognition. They achieved 98.25% and 99.49% recognition results for two different databases. In our experiments on Urdu isolated numerals [45], we used gradient features and achieved high recognition results (97.60%). Gradient features have already been used for word spotting ([49],[50]) in English handwritten texts with encouraging word spotting results.

In our gradient feature extraction phase, after the pre-processing phase, each image of size 128 x 128 pixels was converted back into a greyscale image. The following steps were taken for the extraction of features:

- Robert's filter masks were applied on images. These masks are shown in Fig. 4.8, below:

$$\begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

Fig. 4.8: Robert's Filter Mask for Extracting Gradient Features

- Let I(x, y) as an input image of size 128x128 pixels. After applying the Robert filter masks, the horizontal gradient component (g_x) and vertical gradient component (g_y) were calculated, as follows:

$$g_x = I(x+1, y+1) - I(x, y) \tag{4.1}$$

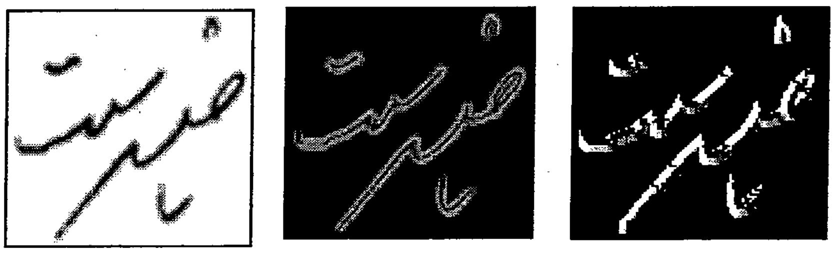
$$g_y = I(x+1, y) - I(x, y+1) \tag{4.2}$$

- The gradient strength and direction of each pixel I(x,y) were calculated as follows:

$$\text{Strength:} \quad f(x, y) = \sqrt{{g_x}^2 + {g_y}^2} \tag{4.3}$$

$$\text{Direction:} \quad \theta(x, y) = \tan^{-1}(g_y/g_x) \tag{4.4}$$

Some examples of gradient images are shown in Fig. 4.9, below:



(a) (b) (c)

Fig. 4.9 : (a) Greyscale image of size 128x128, (b) Gradient Strength, and (c) Gradient Direction

Once the gradient strength and direction for each pixel were calculated, the following steps were taken in order to calculate the feature vector:

I. In equation 4.4, θ(x,y) returns the directon of a vector (g_x, g_y) in the range of [π, -π]. These gradient directions were quantized to 32 intervals of π/16 each.

II. The gradient image was divided into 81 blocks, with 9 vertical blocks and 9 horizontal blocks, as shown in Fig. 4.10. For each block, the gradient strength was accumulated in 32 directions. By applying this step, the total size of the feature set in the feature vector will be (9 x 9 x 32) = 2592.

III. In general, the recognition system would take a lot of time and memory to compute 2592 features from each image in the training and testing sets. If we downsample the feature size, then the system will take less memory and the speed for the training and testing would increase. The trade-off for this benefit of improved time and space is that the recognition rate will most likely decrease.

To reduce the size of a feature vector, a 5 x 5 Gaussian filter was applied to reduce the spacial resolution by down sampling the number of blocks from 9 x 9 to 5 x 5. Also, the number of directions was reduced from 32 to 16 by down sampling with a weight vector [1 4 6 4 1], to obtain a feature vector of size 400 (5 horizontal blocks x 5 vertical blocks x 16 directions).

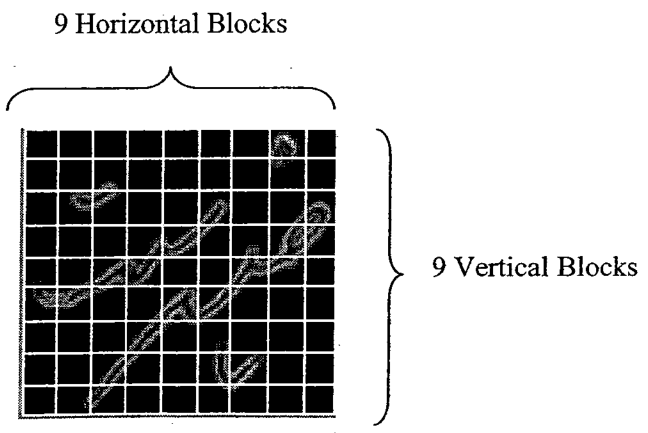


Fig. 4.10: Division of Gradient Image into 9-by-9 Blocks.

IV. A variable transformation (y = $x^{0.4}$) was applied on all features to make the distribution of features guassian-like [36].

We also tested the Sobel filter for extracting the gradient features, but we achieved a higher accuracy with Robert's filter. We will compare both filters in the next subsection.

4.3.2 Robert's Filter vs. Sobel Filter

The gradient can also be computed by the Sobel operator [67], which has two masks for computing the horizontal and vertical gradient components, as shown in Fig. 4.11, below:

-1	0	+1
-2	0	+2
-1	0	+1

+1	+2	+1
0	0	0
-1	-2	-1

Fig. 4.11: Sobel Masks for Extracting Gradient Features.

The horizontal gradient component (g_x) and vertical gradient component (g_y) are computed as follows:

$$g_x = I(u-1,v+1) + 2I(u,v+1) + I(u+1,v+1) - I(u-1,v-1) - 2I(u,v-1) - I(u+1,v-1) \quad (4.5)$$

$$g_y = I(u-1,v-1) + 2I(u-1,v) + I(u-1,v+1) - I(u+1,v-1) - 2I(u+1,v) - I(u+1,v+1) \quad (4.6)$$

In our experiments, we extracted the gradient features by using the Sobel filter and compared our results with the Robert filter. There was not much difference in the results but we achieved a slightly improved performance with the Robert filter. The comparison in experimental results of both filters is shown in Table 4.1, below:

Table 4.1: Robert's Filter vs. Sobel Filter

Image Size	Robert Filter	Sobel Filter	Upper Profile	Results
64-by-64		✓	✓	94.4032% (3559/3770)
64-by-64	✓		✓	94.9602% (3580/3770)
128-by-128		✓	✓	95.7294% (3609/3770)
128-by-128	✓		✓	**96.02% (3621/3770)**

4.3.3 Structural Feature Extraction

The structural features are physical attributes of an image. The structural features include projection profile, topological features [67], upper and lower profiles [48], and so on. The outline shape of a handwritten or printed text is captured by the upper profile and lower profile features [48]. The upper profile features are used to capture the outline shape of the top part of the word, and the lower profile features are used to capture the outline shape of the bottom part of the word. The lower and upper profile features capture almost the same shape of a word if there is no overlapping in the word image. In the Urdu language, overlapping is an issue but in our word dataset, some word classes do not have many overlapping problems. So at this stage in our study, we extracted only the upper profile features from each normalized (64 x 64 size) binary image. The outline shape of a word captured by the upper profile is shown in Fig. 4.12(b), below:

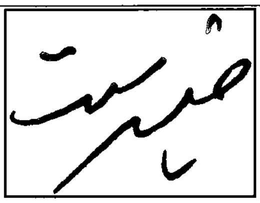
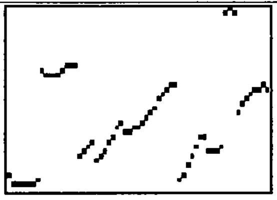
	
(a) Urdu Word Image	(b) Outline Shape

Fig. 4.12: (a) Urdu Word, and (b) Outline shape by using the Upper Profile of (a)

We took the bounding box of the word image so that there was no white space around the handwritten text, and this image was generated in the pre-processing phase. During this process, the following steps were taken to extract the upper profile features:

- The image was converted into a two-dimensional array.
- In each image, each column was examined from right to left.
- In each column, the distance was measured from the top of the image to the closest ink pixel by counting the number of white pixels. This step would return the value of the distance in the range of 0 to 64. This distance calculation from the top of image to the closest ink pixel by counting the number of white pixels is shown in Fig. 4.13, below.

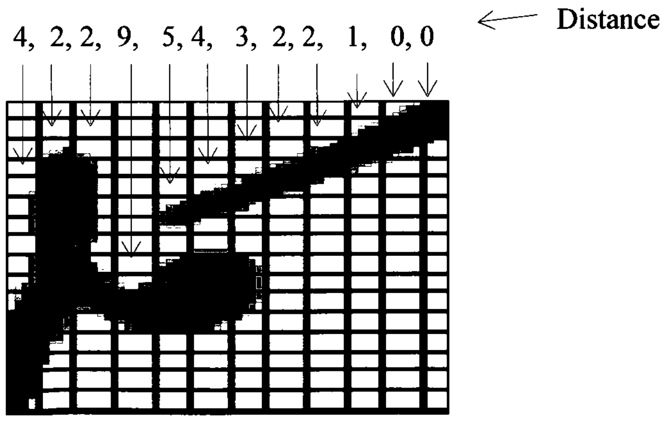


Fig. 4.13: Calculation of Distance (Upper Profile) in Each Column in a Small Portion of the Image

- A variable transformation (y = $x^{0.4}$) was also applied on these features to make the distribution of features guassian-like and to make all the features of the same type and in the same scale.

4.3.4 Feature Vector

After extracting the gradient features and upper profile features from each image, both features were merged together to make a feature vector of size of 464 (400 gradient features and 64 upper profile features). Then, this feature vector was passed to the classification phase. Below, Fig. 4.14 shows the entire feature vector extraction phase.

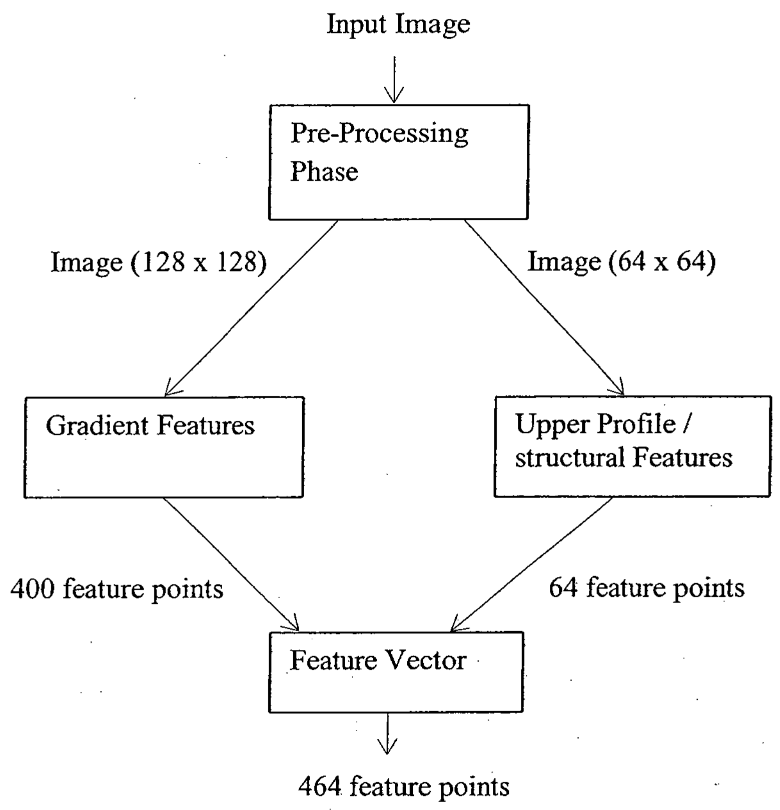


Fig. 4.14: Feature Vector Extraction Phase

4.4 Classification/Recognition

As discussed in chapter 2, the HMM classifier has been the state-of-art classifier for Arabic/Farsi word recognition. In our study, we used the Support Vector Machine (SVM) classifier for recognition purposes. SVM has been accepted as a tool for implementing pattern classification and it has been very successful for offline and online cursive handwriting recognition ([53],[41],[45],[55]) and mostly for character recognition [57]. In 2006, B. Gatos et al. [53] used SVM for cursive handwritten word recognition. Their databases consisted of a training set (23,171 word images) and a testing set (3799 word images). They achieved 87.68 % recognition results while adopting a holistic approach.

The Support Vector Machine is a technique in the field of statistical learning theory. It was primarily designed for 2-class classification problems and is an alternative to Neural Networks. SVM is better than neural networks because of its learning capacity and its structural behaviour.

An SVM performs classification by creating an N-dimensional hyperplane that optimally separates the data into two categories. A brief review of simple binary SVM classification is given in the next section.

4.4.1 Binary SVM classification

In binary SVM classification:

Given is the L dimensional training data (x_1, y_1) (x_2, y_2) (x_3, y_3),... (x_l, y_l) where $x_i \in \mathbb{R}^l$ and $y_i = \{-1,1\}$. If this training data is linearly separable, then a hyperplane or decision factor can easily separate the data into two classes with the help of support vectors. The decision factor is given by:

$$w^T x + b = 0 \tag{4.7}$$

where the vector w defines a direction perpendicular to the hyperplane and is given by:

$$W = \sum_{i=1}^{l} \alpha_i \, y_i x_i \tag{4.8}$$

and

α_i = Coefficient weights

b = Constant

A decision factor, as in Eqn. 4.7, helps in maximizing the margin from the hyperplane to the support vector (see Fig. 4.15). An example of simple binary classification is shown in Fig. 4.15, below:

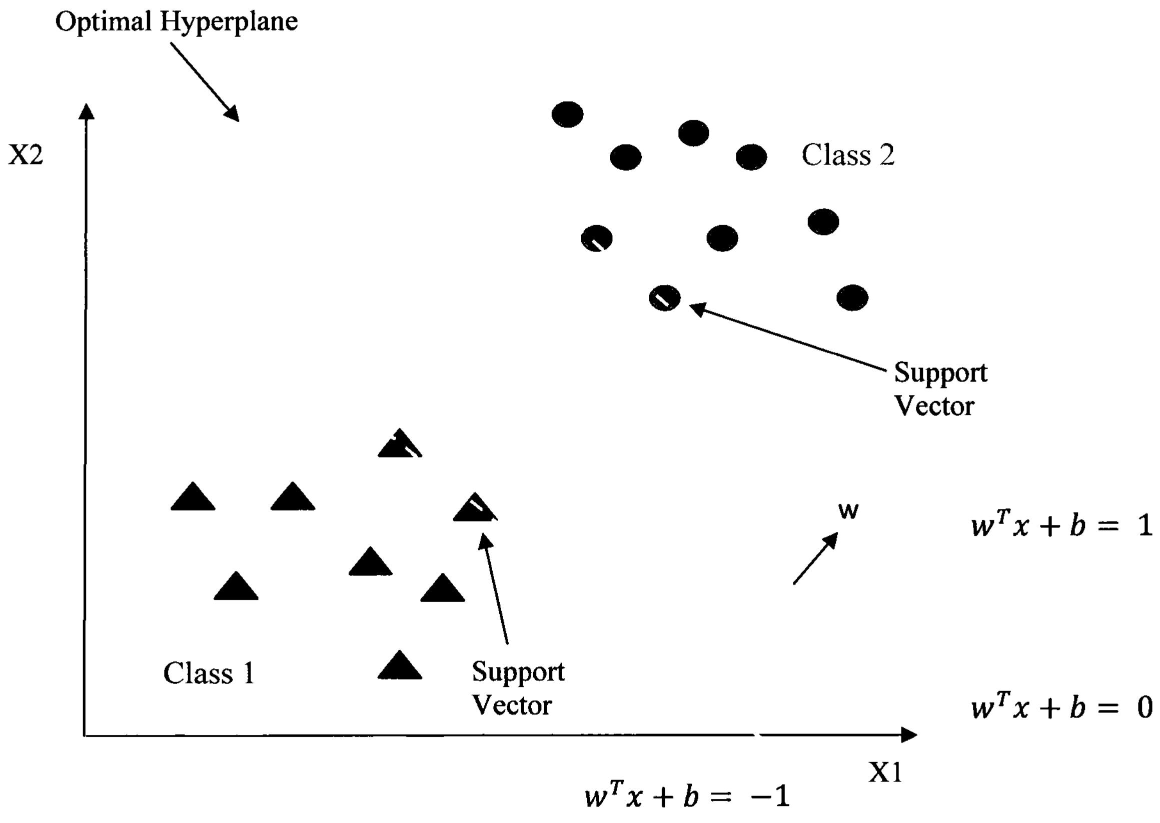


Fig. 4.15: Binary Classifier with SVMs

When training data is not linearly separable, then SVM uses a kernel function (k) to map (Φ) data into a higher dimensional space (feature space), where data can be linearly separated, as shown in Fig. 4.16, below:

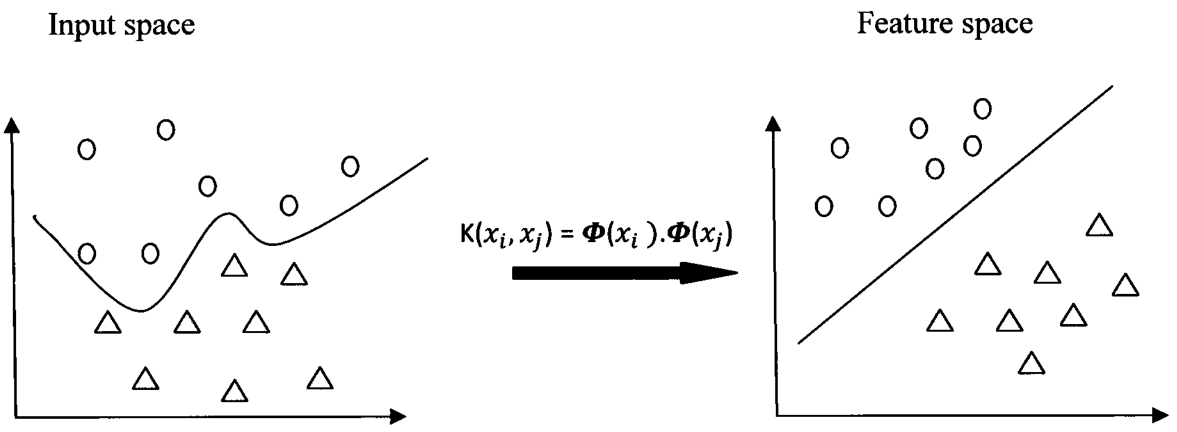


Fig. 4.16: Mapping of data into a higher dimensional space

Detailed explanations of SVM can be found in ([59],[60],[61],[62]).

In our experiments, we used LibSVM, an open source library for the implementation of SVM [56]. LibSVM takes the input feature matrix and outputs the classification results in probabilities. LibSVM is a one-against-one [63] strategy implementation of SVM in multi-class problems. LibSVM uses the Radial Base Function (RBF) kernal for mapping a nonlinear data sample into a higher sample space. The RBF kernal which presents the first norm is given by:

$$K(X_i, X_j) = \exp(-\gamma \|X_i - X_j\|^2), \quad \gamma > 0. \qquad (4.9)$$

For the k-class problem, k^2 SVMs are trained with a pairwise approach and the probability p_i is estimated for any test sample x that belongs to class i. The probabilties are obtained from r_{ij} where (i, j = 1,2,3,..., k). The r_{ij} is a one-against-one class probability and it is obtained from known training data by solving the following problem:

$$min_p \frac{1}{2} \sum_{i=1}^{k} \sum_{j:j \neq 1} \left(r_{ij} P_i - r_{ij} P_j \right)^2$$

where

$$P_i = P(y = i|x), \text{ for class label y of x.}$$

4.4.2 Training

Before training, the two optimal parameters γ and C were chosen by using v-fold cross-validation. In Eq. 4.9, γ is a parameter used in RBF and C is a penalty parameter for the error term. After choosing the best parameters γ and C, we trained our system on the whole training set. In our Urdu word dataset we had 57 classes and in the training folders from each class, we had 11,104 images in total. We provided a matrix of 11,104 feature vectors with their class labels for training purposes.

4.4.3 Testing

In the testing phase, we used the same parameters γ and C as used in the training phase. We tested our system with 3770 images. These images were taken from the testing folder of each class. Our system successfully classified 3621 images. The overall HWR recognition system flowchart is shown in Fig. 4.17, below:

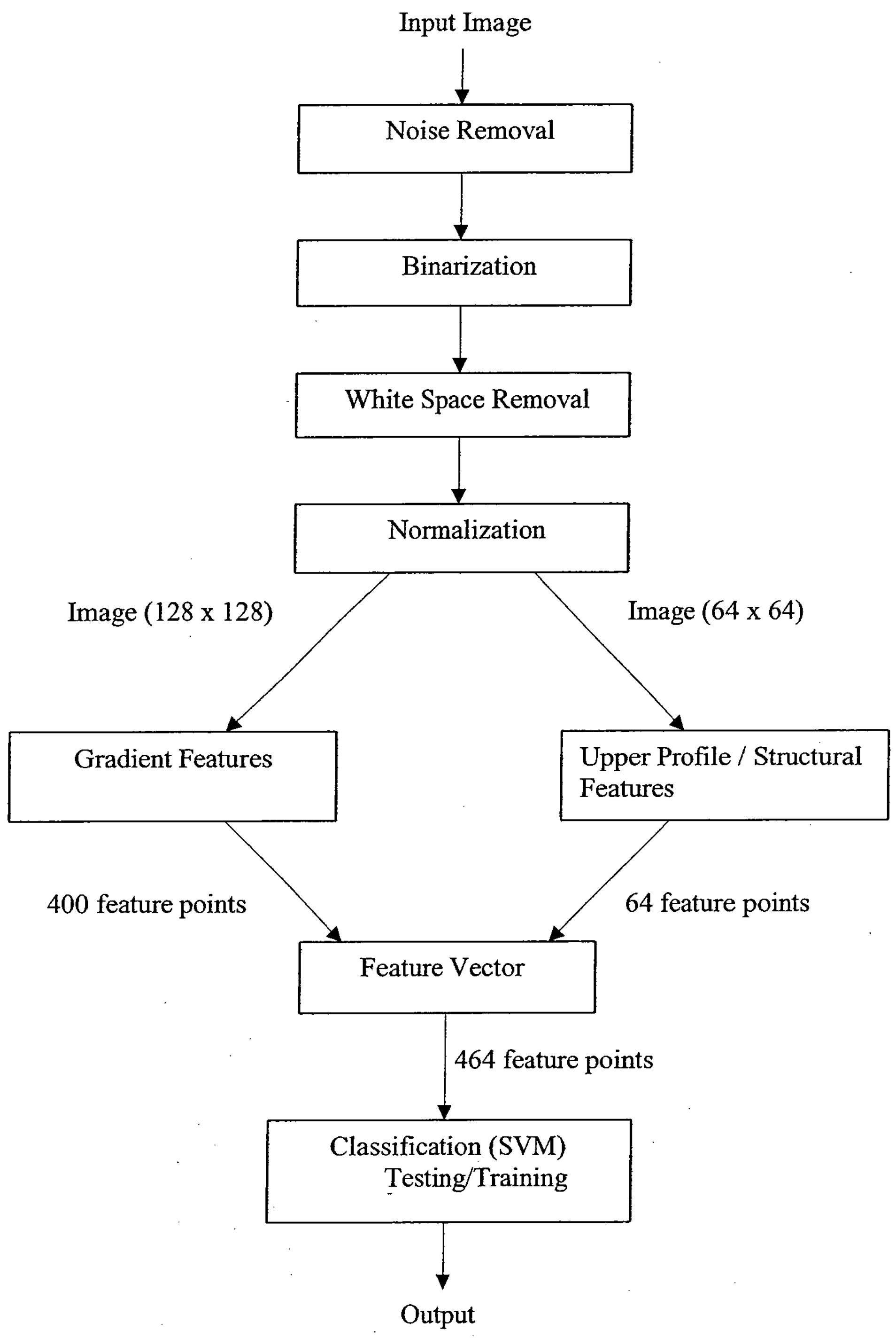


Fig. 4.17: Overall Handwritten Word Recognition System Flowchart

4.5 Experiments and Results

We performed different experiments to obtain the best results. In our experiments, the recognition results varied according to the size of the images for normalization, the different features and the different number of training samples. In the end, we achieved 97.00% recognition rate results with 592 feature points and with an increased size (14,407) of the training set. In the following subsections, we will discuss these experiments, their recognition results and their error analyses.

4.5.1 Image Size for Normalization

Different image sizes were used in our normalization process in order to find the best image size for recognition purposes. Increasing the size of the image also increased the recognition results, but it slowed down the recognition process. We found the best results with the image size of 128 x 128 pixels when extracting gradient features. For upper profile features, the image size of 64 x 64 pixels was found to be optimal for extracting outer shape of a word image. In these experiments, we fixed the size of 64 x 64 pixels for extracting 64 upper profile features and tested different image sizes for extracting the 400 gradient features. The experimental results with different image sizes are shown in Table 4.2, below:

Table 4.2: Experimental Results with Different Image Sizes for Gradient Features

Image Size (Gradient Feature)	Feature Size	Results
64-by-64	464	94.9602% (3580/3770)
64-by-128	464	95.2255% (3590/3770)
128-by-64	464	95.0928% (3585/3770)
128-by-128	464	**96.02% (3621/3770)**

From the above table, it is evident that the best results were achieved by image size 128-by-128. The results could still be improved if the image size is increased but it will slow up the recognition process. The typical errors generated from these experiments will be explained in section 4.5.3.

4.5.2 Improving the Results

In our experiments, our main focus was to improve the recognition results. We found that if we added more suitable features to our feature vector, then the recognition results would improve. Also, we improved our recognition results by adding more images to our training set, to train our system on more samples. We will discuss both of these procedures in the following subsections.

4.5.2.1 Adding More Sets of Features

The recognition result is dependent on the selection of the best features. Different features like horizontal and vertical projection histogram features [8], and structural features were used in our experiments. Fig 4.18 shows horizontal and vertical histograms of an Urdu word. We extracted 64 horizontal features and 64 vertical features from each binary image of size 64 x 64 pixels. In our experiments, we found a better performance with the combination of gradient features, upper profile features and histogram features. The recognition results with different feature sets (fv) are shown in Table 4.3.

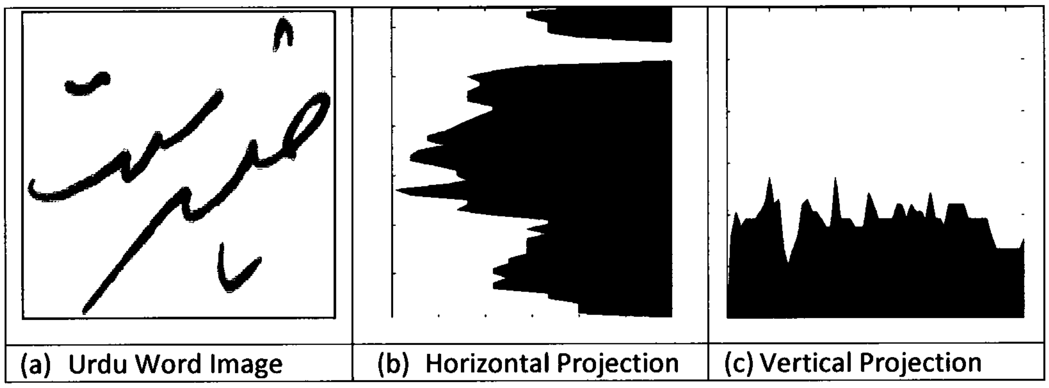


Fig. 4.18: (a) Urdu Word Image, (b) its Horizontal Projection, and (c) Vertical Projection

Table 4.3: Comparison of Recognition Results with Different Features

Feature Vector	**Gradient Features**	**Horizontal Projection**	**Vertical Projection**	**Upper Profile**	**Size of a Feature Vector**	**Recognition Results**
fv1				✓	64	68.14% (2569/3770)
fv2		✓	✓		128	77.24% (2912/3770)
fv3		✓	✓	✓	192	86.36% (3256/3770)
fv4	✓				400	95.41% (3597/3770)
fv5	✓			✓	464	96.02% (3621/3770)
fv6	✓	✓	✓	✓	592	**96.63% (3643/3770)**

In the above table, the first column shows the different feature sets, column two to column five show the different feature types, column six shows the total number of feature points in a feature vector and the last column shows the recognition results for each feature set. Column six and column seven clearly show that the recognition results improve by adding more features.

4.5.2.2 Increasing Training Samples

At first, we were only using the training folder from our CENPARMI Urdu databases for training purposes. As we were not using the validation folder for any purpose, this folder was added to the training folder. The system was trained again on this combination of two folders and we noticed enhanced results for the testing folder. This is the best example for machine learning. When the machine is trained on more data samples, then the machine is able to predict the result more accurately. For our experiments, the results are shown in Table 4.4. In the next subsection, we will focus on recognition errors, to understand the problems faced in word recognition.

Table 4.4: Improved Results by Enhanced Training Set

Training Samples	Test Samples	Features	Recognition Results
11,104 Images	3770 Images	464	96.02% (3621/3770)
14,407 Images	3770 Images	464	**96.87% (3652/3770)**
14,407 Images	3770 Images	592	**97.00% (3657/3770)**

In the above table, the first column shows the total number of training samples, the second column shows the total number of test samples, the third column shows the total number of features and the last column shows the recognition results. The first column and the last column show that the recognition results improve by adding more training samples. Finally, a recognition result of 97% was achieved by adding more feature sets.

4.5.3 Error Analysis

There are a few words in our CENPARMI Urdu database that are visually similar to each other and our system gave most of the wrong results for those images, and we found the wrong output for those word images in the recognition process. For example two Urdu words, translated into English as “Litre” and “Metre”, look very similar and are shown in Fig. 4.19, below:

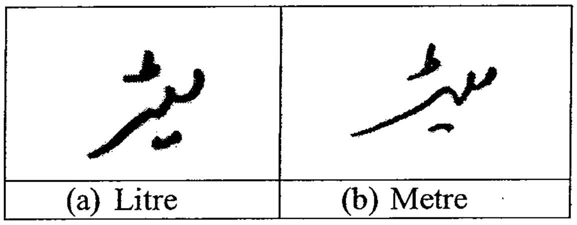


Fig. 4.19: Two Similar Words in the Urdu language

Some other similarities in shapes of different Urdu words are shown in Fig. 4.20, below:

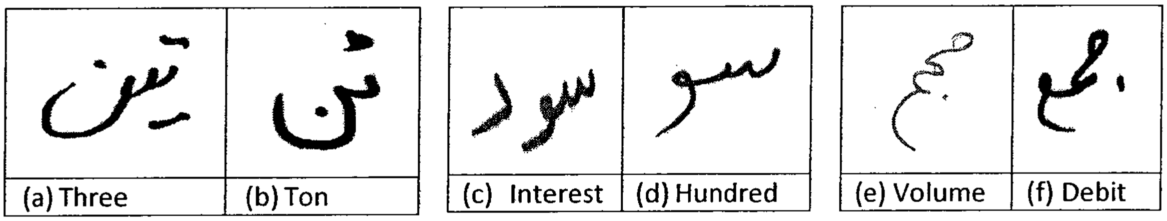


Fig. 4.20: Similarities in Urdu Words

The complete error analysis can be seen in a confusion matrix in Appendix B.

4.6 Comparison of a System with Existing System

In past research on Arabic/Farsi offline word recognition, we noticed that the Hidden Markov Model (HMM) was the default choice for many researchers

([4],[9],[19],[21],[25]). To the best of our knowledge, this is the first time an SVM classifier is being used for offline Arabic (Urdu) word recognition. In January 2010, P. J.-Haghighi [69] at Concordia University, Montréal used the Discrete HMM for the recognition of Farsi words. She used the CENPARMI Farsi word dataset [40], which consists of 73 Farsi words. In this system, recognition was performed on a word level (Holistic-based). For comparison with that system, we trained and tested our system on the same CENPARMI Farsi word dataset and achieved almost the same recognition accuracy on the test set. The training set consisted of 20,817 images and the testing set consisted of 7007 images. A comparison between these two systems is shown in Table 4.5 below:

Table 4.5: Comparison of Recognition Results on CENPARMI Farsi Word Testing Set

Researcher	Classifier Used	Number of Features	Recognition Results
P.J.Haghighi	Discrete HMM	75,000 (aprox)	96.04%
M.W.Sagheer	SVM	592	95.70%

The second column of Table 4.5 shows the classifier used for each system. The third column shows the sizes of the feature vectors and the fourth column shows the recognition results. Both systems produced almost the same results. Our system consisted of only 592 feature points and we already showed in Table 4.3 that the recognition results rose after increasing the size of the feature vector. However, extracting 75,000 features from each image will slow the recognition process. We could not compare the overall recognition process time because the recognition performance time was not available in [69]. Overall, we achieved a 95.70% recognition result which is very close to the result of

Haghighi. These results show that the SVM classifier can be very useful for the recognition of Arabic, Farsi or Urdu words.

4.7 Conclusion

In this chapter, we described our handwritten word recognition system. We described different pre-processing techniques and feature extraction processes. We noticed that the recognition results improved by adding more feature sets and more training samples. Usually, the SVM classifier is not preferred for an Arabic word recognition system but we used SVM, and obtained high recognition results. We also compared our system with an HMM-based system and achieved similar results. In the future, our system may be extended by adding new words.

As mentioned in Chapter 1, word recognition can be used for different tasks. From those tasks, one of the best examples is word spotting on handwritten text pages. There has been very minimal work done on Arabic word spotting because of the complexity in the segmentation of handwritten text pages and lines. We applied our HWR system to word spotting and it will be discussed in the next chapter.

CHAPTER 5

Urdu Word Spotting Technique

5.1 Introduction

Handwritten word spotting has already gained significant attention among researchers in the field of Pattern recognition. In Arabic languages, different people write texts in different styles. There is no defined method to write two words. Some people could overlap these two words, while others may give enough space between two words, and still others may give more space within a word (inner-word space) than the space given between two words (outer-words space). This situation poses a problem and makes word segmentation a very difficult process as a wrong segmentation can lead to the wrong word spotting. An example of overlapping and variation in an inner-word and space between two words in an Urdu handwritten text line is shown in Fig. 5.1, below:

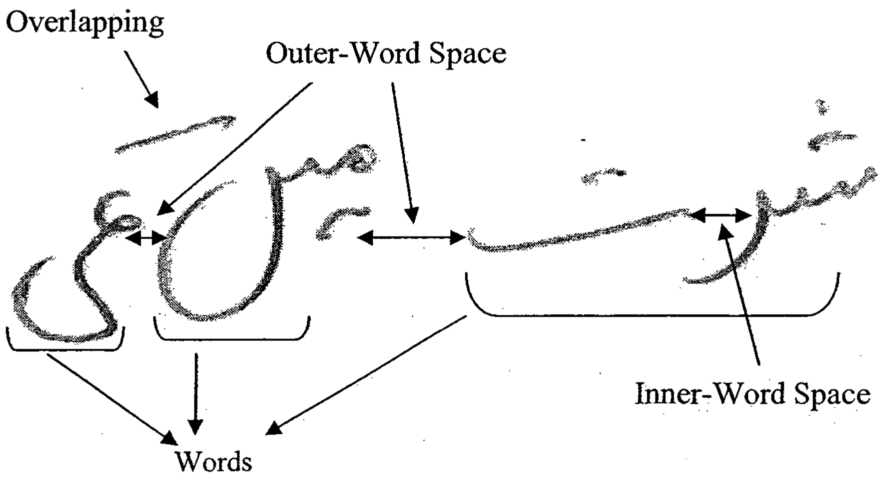


Fig. 5.1: A Variation of Inner-Word and Outer-Word Space

Our proposed word spotting system was implemented on text lines and it consists of the following phases:

- Handwritten Text Databases
- Pre-processing
- Text Line Segmentation
- Feature Extraction
- Word spotting

We will now discuss each phase in the proceeding topics.

5.2 Handwritten Text Databases

For our word spotting experiments, a handwritten Urdu text database was not available. For this reason, we first collected a small Urdu database at CENPARMI, in Montreal, at Concordia University. This database consisted of a total of ten writers, all of whom were native Urdu speakers. Each writer was asked to write four pages and approximately ten lines of text on each page. All samples were taken from our CENPARMI Urdu databases and each sample consisted of isolated digits, numeral strings, dates and words. We added fifty-seven words, from our CENPARMI Urdu database, in different text line positions. The writers were also asked to write in a free style on plain pages. All of these writers did not participate in creating the CENPARMI Urdu databases. So, for word spotting experiments, we used writing samples from these ten new writers. A sample of a handwritten Urdu text page is shown in Fig. 5.2.

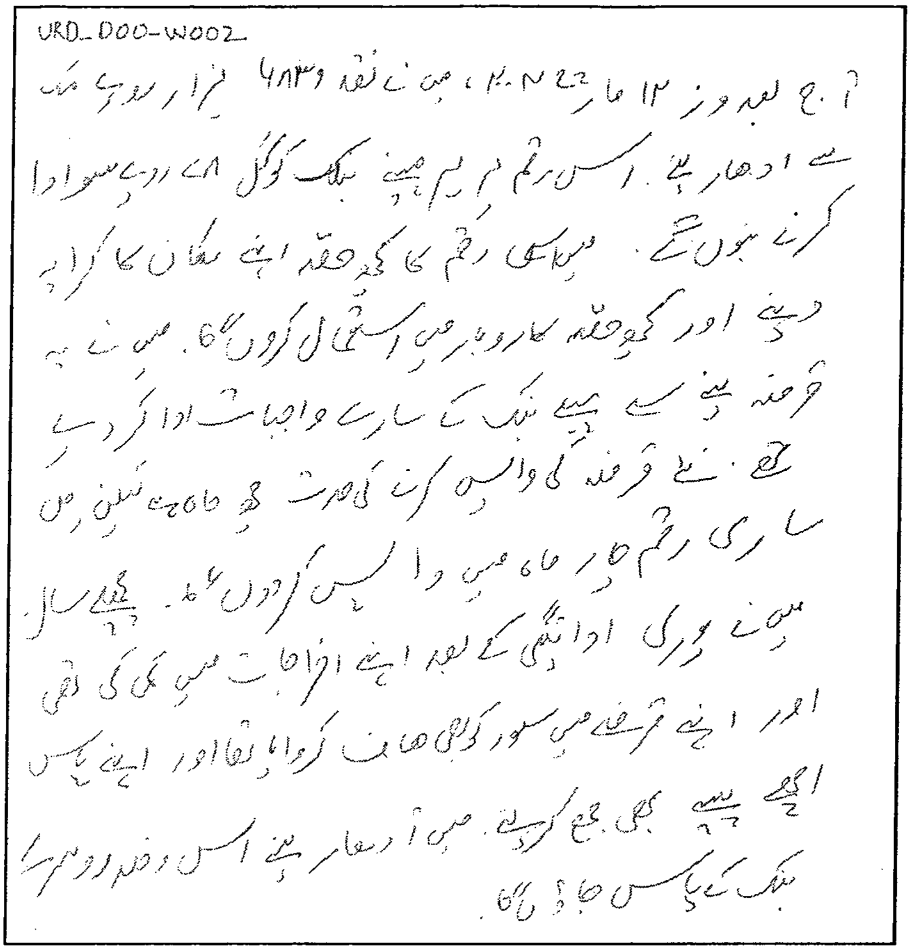
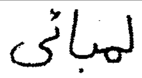
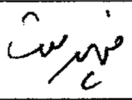
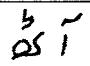
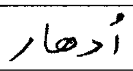
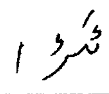
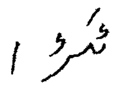
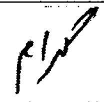


Fig. 5.2: Sample of a Handwritten Urdu Page

5.3 Word Segmentation

In our word spotting system, we assumed that the text pages were segmented into text lines. Firstly, we manually extracted lines from each text page. Then, each line was segmented into main connected components (CCs) with their secondary components, also known as diacritics, in order to find the words. Diacritics appeared above and below the main components. There are nine different diacritics in the Urdu language, and they are shown in Table 5.1.

Table 5.1: Nine Different Urdu Diacritics

Diacritic name	Image	Words
Single Dot		
Double Dot		
Hamza		
Hey		
Mudd		
Pesh		
Small Toway		
Single line for kaaf		
Double lines for Gaaf		

These diacritics are the most important parts of words and their absence/position can change the meaning of words. Some isolated characters may have the same main component shapes but they differ from each other by their diacritics. In Table 5.2, there are some isolated characters with the same main component shape but with different diacritics in different positions.

Table 5.2: Similar Main Component Shapes with Different Diacritics

ث	ٹ	ت	پ	ب
Se	**Tay**	**Te**	**pey**	**Be**

In Urdu, a combination of two or more characters makes a word or part of a word. In a text line, each connected part of a word is called a main connected component (CC) or a diacritic, as shown in Fig. 5.3, below:

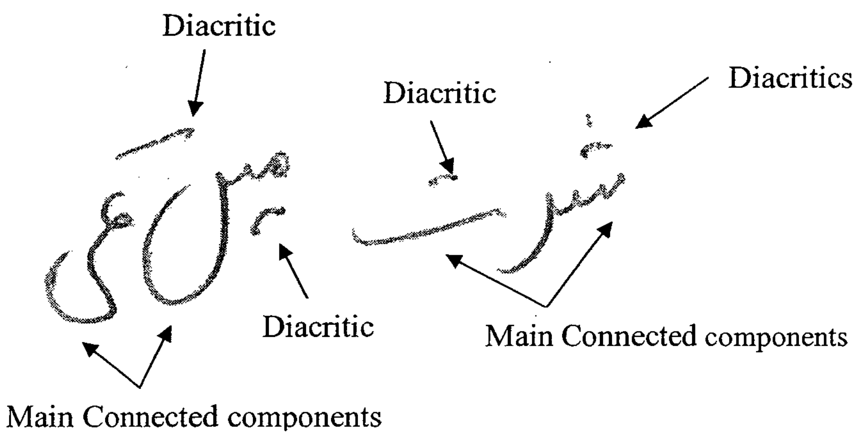


Fig. 5.3: Main Connected Components and Diacritics in an Urdu Text Line

In the handwritten word segmentation, our goal was to group the main CC's with each of their diacritics and then segment those groups. Grouping the main CC's with their diacritics is not an easy task. Diacritics are usually written above or below the main CC. Their geometrical features cannot guarantee that they are recognized as diacritics or that each is grouped with its own main component. Also, overlapping of two words is common and sometimes the size of a diacritic is bigger than the main CC. This overlapping or the size of diacritics can sometimes cause the wrong grouping. Also, at times diacritics touch the baseline. Because of these problems, the segmentation of a text line is difficult. As we only segmented the main connected components, we first had to recognize the diacritics and their main components so that each main connected component could be segmented properly. In the following subsection, we will discuss our pre-processing of a text line and our segmentation algorithm.

The upper and lower baselines in a handwritten Urdu text are shown in Fig. 5.4, below:

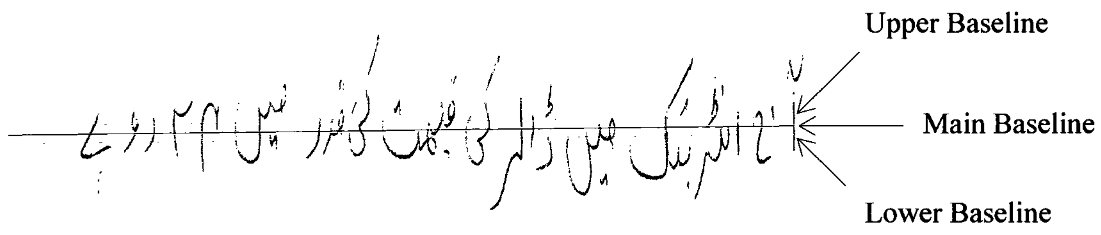


Fig. 5.4: Upper and Lower Baselines in a Text Line

5. Each connected component in the binary image was labeled.

5.3.2 Segmentation Algorithm

After the pre-processing of a text line, we performed the following steps to group the diacritics with their main components:

1. Connected components that were coming above the upper baseline and below the lower baseline were considered as the diacritics.
2. The skeleton of each connected component, which touched the baseline area (between upper and lower baselines), was obtained by using the Zhang-Suen thinning algorithm [64]. This algorithm takes the Boolean image and reduces it to an 8-connected skeleton. In this algorithm, each iteration consists of two steps. The southeast boundary points and northwest corner points are removed in the first step ('0' pixel values are replaced by '1'). The northwest boundary points and southeast corner points are also removed in the first step. Let p_1, p_2,….,p_8 represent eight neighbour pixels of point p as shown in Fig. 5.5, below:

p_8	p_1	p_2
p_7	p	p_3
p_6	p_5	p_4

Fig. 5.5: The Eight Neighbours of pixel p

Let B(p) be the number of nonzero eight neighbour pixels of pixel p and let A(p) be the number of zero-to-one conversions in the ordered sequence $p_1, p_2, \ldots, p_8, p_1$. If p is a contour point (at least one pixel value out of the eight neighbour pixel values of p is equal to '0') then in step one, then the following four conditions should be satisfied in order to flag the pixel p for removal:

1. $2 \leq B(p) \leq 6$
2. $A(p) = 1$
3. $p_1 \cdot p_3 \cdot p_5 = 0$
4. $p_3 \cdot p_5 \cdot p_7 = 0$

The removal of pixel p will be delayed until all the pixels of the image have been examined by the algorithm. In the second step, the contour point p is flagged for removal if the following four conditions are satisfied:

1. $2 \leq B(p) \leq 6$
2. $A(p) = 1$
3. $p_1 \cdot p_3 \cdot p_7 = 0$
4. $p_1 \cdot p_5 \cdot p_7 = 0$

After applying the above two steps to all the pixels of the image, the resulting image will be the skeleton. An example of a skeleton that was generated from the Urdu word image is shown in Fig. 5.6, below:

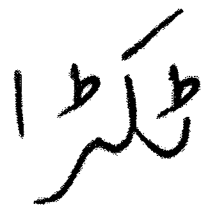
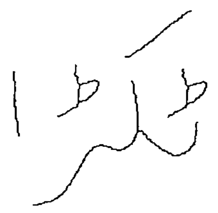
	
(a) Urdu Word	(b) Word Skeleton

Fig. 5.6: (a) Urdu Word image, and (b) its Skeleton generated by the Zhang-Suen Algorithm

The number of black pixels of each skeleton was counted to identify the diacritics. As the skeletons of dots had a maximum of four pixels, the process of finding dots was usually easy. We had a threshold value of 50, to identify the diacritics by their size which was given by:

$$\text{Threshold value} \leq 50 \text{ pixels} \qquad \text{(for Diacritics)}$$

The value 50 (number of black pixels) was found to be the best threshold value for diacritics in our experiments on the CENPARMI Urdu word training sets.

3. Sometimes the size of the diacritics was more than 50 pixels. We considered them as main connected components and saved them as separate images.

   Once all the diacritics were located, the grouping process was performed. An Urdu handwritten text line with its main CC's and diacritics are shown in Fig. 5.7 and the diacritics from that text line are shown in Fig. 5.8.

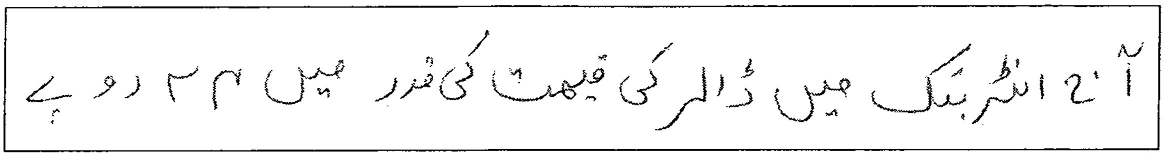


Fig. 5.7: Urdu Handwritten Text Line

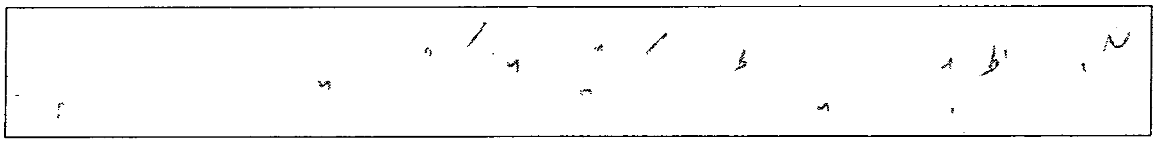


Fig. 5.8: Diacritics Found in the Text Line of Fig. 5.7

4. For each recognized diacritic, we located its nearest main connected component (base) and saved both the base component and its associated diacritic as a separate image. We also saved a record of the original location of that component on the text line. We then extracted the main connected components (CC) with their diacritics from right to left. We took a text line, as in Fig 5.7, and then extracted all the main CC's with their diacritics from this text line, as shown in Table 5.3, below:

Table 5.3: Segmented Connected Components With Their Diacritics

CC.13	CC.12	CC.11	CC.10	CC.9	CC.8	CC.7	CC.6	CC.5	CC.4	CC.3	CC.2	CC.1

The overall handwritten word segmentation system is shown in Fig. 5.9, below:

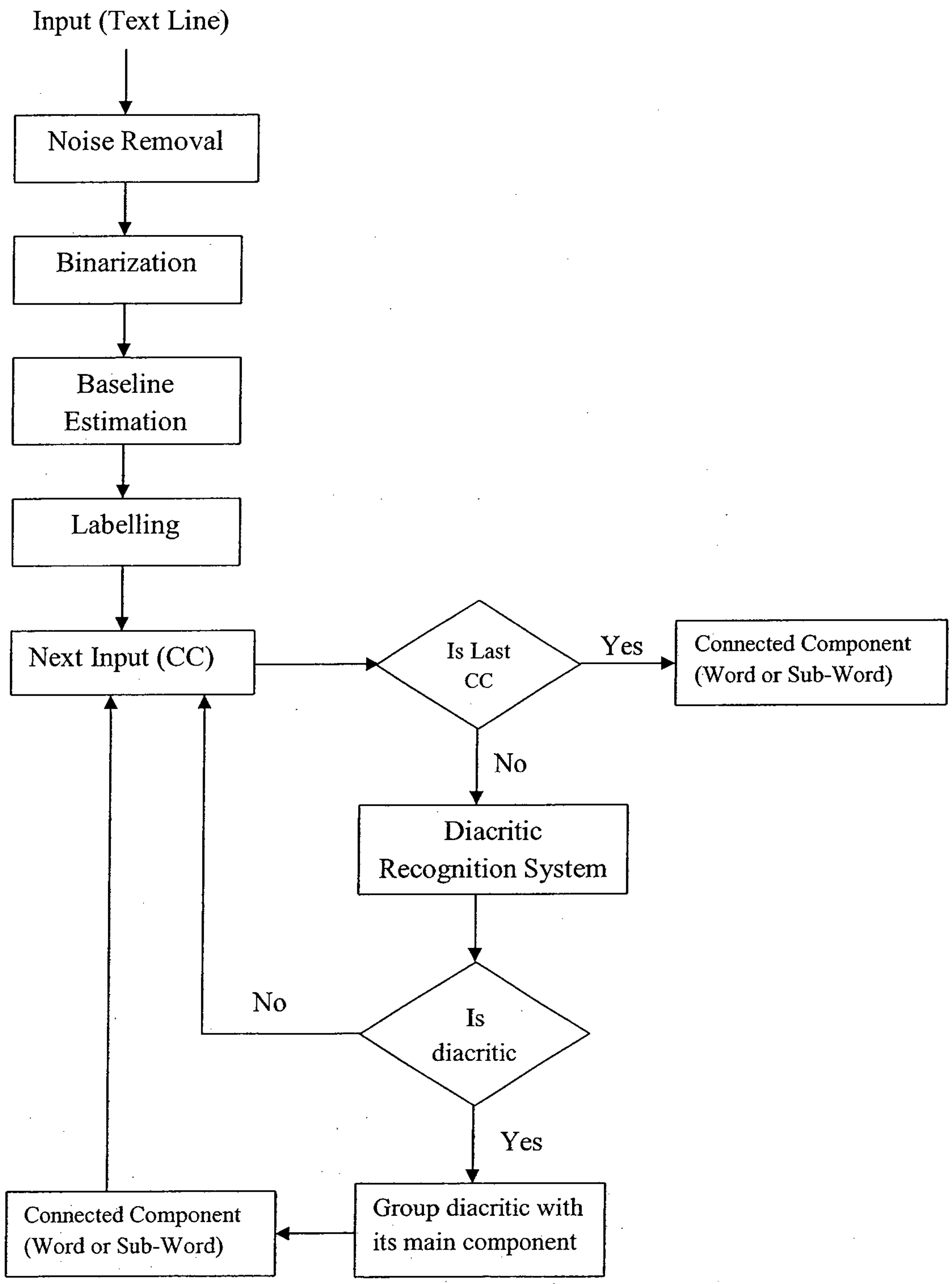

Input (Text Line)
Noise Removal
Binarization
Baseline Estimation
Labelling
Next Input (CC)
Is Last CC
Yes
Connected Component (Word or Sub-Word)
No
Diacritic Recognition System
No
Is diacritic
Yes
Group diacritic with its main component
Connected Component (Word or Sub-Word)

Fig. 5.9: Word Segmentation Flowchart

5.3.3 Segmentation Results

We tested our segmentation method for each handwritten text line from 10 documents written by ten different writers. A few of those writers did not write clearly and there was much overlapping in their handwritten texts. But we still obtained satisfactory segmentation results. The segmentation results for each writer are given in Table 5.4, below:

Table 5.4: Segmentation Results

Writer Number	**No. of text Lines**	**Total Segmented Subwords**	**Correctly Segmented subwords**	**Correct Segmentation rate**
W0000	37	888	823	92.68%
W0001	32	809	743	91.84%
W0002	38	846	766	90.54%
W0003	40	803	731	91.03%
W0004	36	865	780	90.17%
W0005	39	857	803	93.70%
W0006	45	875	835	95.43%
W0007	38	909	850	93.51%
W0008	34	872	817	93.69%
W0009	33	829	730	88.06%
Overall	**372**	**8553**	**7878**	**92.11%**

In Table 5.4, the first column shows a number assigned to identify each writer and the second column shows the total number of text lines written by each writer. The third column shows the total number of segmented subwords (main CC's with their diacritics) written by each writer. The fourth column shows the total number of correctly segmented subwords (main CC's with their diacritics) and the last column represents the correct segmentation rate for each writer.

Overall, we tested our segmentation method on 372 text lines, and we achieved a 92.11% correct segmentation rate overall. We will discuss some segmentation errors in the following subsection:

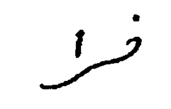
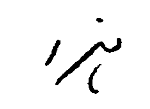
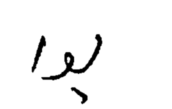
5.3.4 Segmentation Error Analysis

There were a few major segmentation errors which could have resulted in wrong word spotting but overall, the segmentation errors were insignificant and may be overcome in future research. Most of these errors came from the Alif (ا) character and also, from the two diacritics: Single line for kaaf (/) and double lines for kaaf (⫽). We will review these errors in the following subsections.

5.3.4.1 Errors with Alif

In our segmentation algorithm, we were calculating the number of black pixels and the position of each CC to differentiate between the main CC and the diacritics. Most of the writers wrote the Alif (ا) character too small and its number of black pixels was less than 50 pixels, which was the threshold value for diacritics in our system. We found that if Alif (ا) character was coming in the middle or at the ending position in a word, then this Alif character was overlapped on top of the previous main CC and it was also not touching the baseline area. Because of these problems, most of the time the Alif character was recognized as the diacritic and was merged with its previous main CC. Some examples of these segmentation errors are shown in Table 5.5, below:

Table 5.5: Segmentation Errors with the Alif character

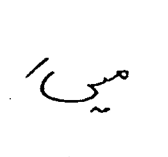
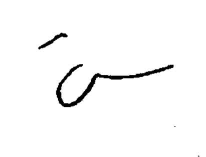
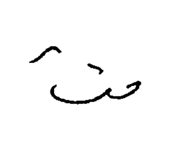
		
(a)	(b)	(c)

In the above table, the images in a, b and c contain the Alif character, but the Alif characters are incorrectly segmented from the main CC's.

5.3.4.2 Errors from Diacritics

Most of the diacritics were merged correctly with their main CC in our segmentation method, except for two: single line for kaaf (/) and double lines for kaaf (⫽). The two diacritics are alike and have the same geometrical information. Both of the diacritics are always written on top of the main CC. We found that these diacritics were overlapped with two main CCs (their own main CC and the previous main CC). In our segmentation method, text lines are processed from right to left and these two diacritics were merged with the closest main CC (the previous main CC) instead of their own main CC. Some examples of these segmentation errors are shown in Table 5.6, below:

Table 5.6: Segmentation Errors with Single line for kaaf (diacritic)

		
(a)	(b)	(c)

In the table above, the images in a, b and c contain the single line for kaaf diacritic but these diacritics are incorrectly segmented from the previous main CC's.

5.4 Word Spotting

Our word spotting method consisted of the following phases:

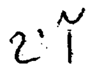
- Word generation
- Word spotting algorithm
- Verification/Rejection

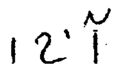
We will discuss each phase in the following subsections:

5.4.1 Word Generation

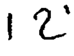
Because there is variation in the outer-words space in a handwritten Urdu text, it is not possible to know how many words can be segmented in a handwritten text line. The segmentation of a text line into connected components gave us the option to make the word by combining neighboring components together. Given a text line as in Fig 5.7, segmented into connected components as shown in Table 5.3, then from the first three connected components (CC.1, CC.2, CC.3), the candidate words [65] can be generated, as shown below:

First candidate word: (CC.1)

Second candidate word:  (CC.2 + CC.1): read from right to left

Third candidate word:  (CC.3 + CC.2 + CC.1)

Fourth candidate word: (CC.2)

Fifth candidate word:  (CC.3 + CC.2)

Sixth candidate word: (CC.3)

Suppose there are n connected components in a given text line, then the number of possible Candidate Words (CW) is given by:

$$\text{CW} = \frac{n(n+1)}{2} \tag{5.1}$$

For our research, we only extracted words with four or less connected components. Out of fifty-seven classes (words) in our CENPARMI Urdu databases, fifty-six consisted of 4 or less connected components. In our word spotting algorithm, we used a sliding window to extract the candidate words using the following algorithm:

If there are n connected components in a text line, then the size of a sliding window (*SW*) at each iteration is given by:

$$SW = 4 \quad for\, n \geq 4 \tag{5.2}$$

$$SW = n \quad for\, n < 4, \tag{5.3}$$

and the total word regions (*R*) on a text line are given by:

$$R = n \tag{5.4}$$

So there will be n iterations to extract the candidate word regions in any given text lines. An example of candidate word regions in a text line is shown in Fig. 5.10, below:

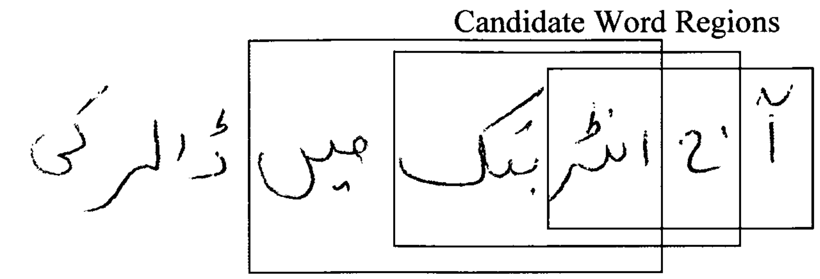


Fig. 5.10: Candidate Word Regions in a Text Line

In each word region, we can generate up to four words by combining neighbouring CC's together from right to left. In a text line, there will be (R – 3) regions having four CCs and we can generate 4*(R-3) candidate words from those regions. From the remaining three regions, we can generate 6 candidate words. In this case, the total number of Candidate Words (CW) in the text line is given by:

$$\mathrm{CW} = 4(R-3)+6 \qquad for\, n \geq 4, \text{ and} \qquad (5.5)$$

$$\mathrm{CW} = \frac{n(n+1)}{2} \qquad for\, n < 4, \qquad (5.6)$$

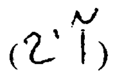
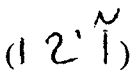
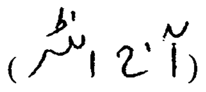
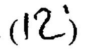
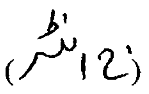
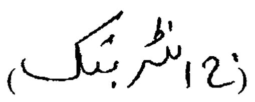
where $R = n$ (Eqn. 5.4) and n = the total number of connected components.

By using this method, we covered all the possibilities to check every individual component and to check the union of neighbouring components up to a maximum of three inner-word spaces. It was sometimes a slow process to recognize each of these components, and the unions of these components.

After generating the words, each candidate word was sent to a classifier for classification, as explained in the next subsection.

5.4.2 Word Spotting Algorithm

We used the following algorithm to spot (recognize) the words:

1. In the first iteration, the following four images were sent to the classifier:
   a. CC.1
   b. CC.2 + CC.1 
   c. CC.3 + CC.2 + CC.1 
   d. CC.4 + CC.3 + CC.2 + CC.1 
2. In the second iteration, the following four images were sent to the classifier:
   a. CC.2
   b. CC.3+ CC.2 
   c. CC.4 + CC.3 + CC.2 
   d. CC.5 + CC.4 + CC.3 + CC.2 

We used our holistic HWR system for recognition purposes, as explained in the next subsection.

5.4.2.1 Classification

Classification in the word spotting algorithm is different from the one described in chapter 4. In this case, most of the time the words that we tried to recognize did not exist in our database. We wanted the system to only try to recognize the words that existed in the databases and to reject all the other words.

As discussed earlier, LibSVM generates the outputs for test samples in probabilities. On the basis of these probabilities, the system recognizes the class for that sample. Given that a system is trained on three classes, the classification of any sample point (I) is given by the following formula:

- The system will first extract the feature vector.
- The system will then map that feature vector into the Feature space.
- In the feature space, the system will find the closest hyperplane of Class N, which can divide all samples into two classes (*is* and *isn't*), and assign this test point to Class N. For example, a classification of test sample I with three classes is shown in Fig. 5.11, below:

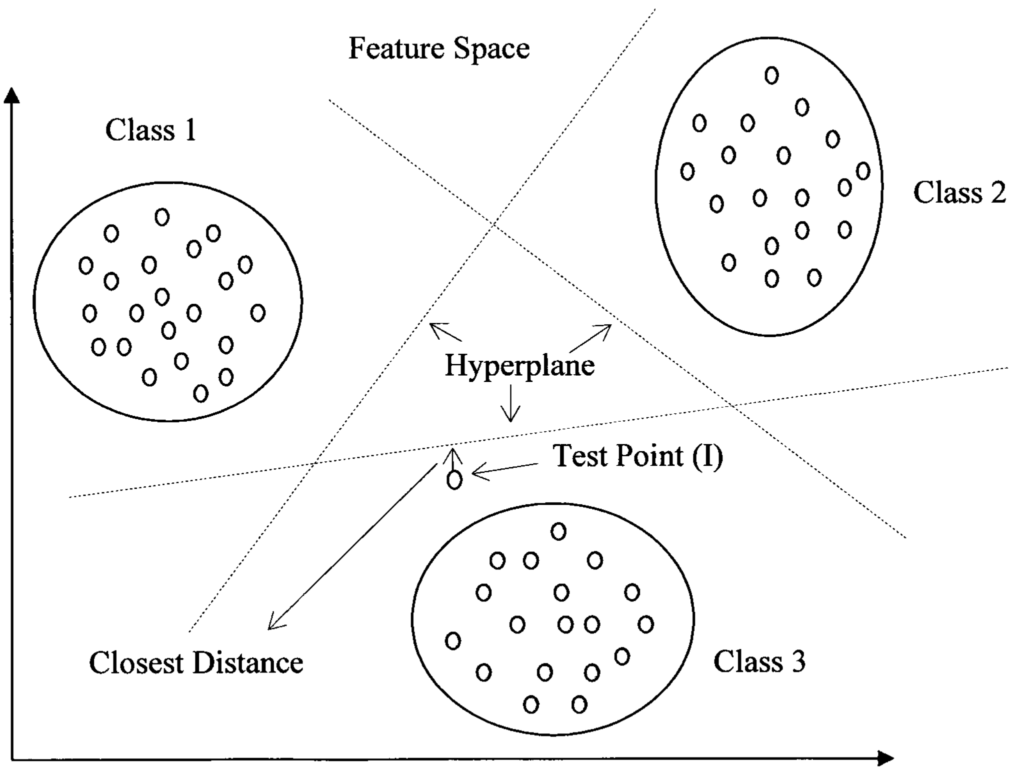


Fig. 5.11: An Example of Prediction of Sample Test Point (I) with Three Classes in a Feature Space

For this example, LibSVM generates the results as follows:

Sample	Class 1 (P)	Class 2(P)	Class 3 (P)
I	0.1	0.2	0.7

Where P is the probability

If the class of a sample point does not exist in the database or SVM is not trained on that class, then SVM will map that sample into the feature space as it does for known data, and it will try to classify that sample to a closest class. These kinds of errors are called *false positive.*

In our system, after segmentation, each image went through the following processes for classification:

- Pre-processing: As the image was already binarized and noise removal was already performed on the text line, we only normalized the image into 64-by-64 and 128-by-128 sizes.
- 400 gradient features [36] were extracted from the greyscale image of size 128-by-128.
- 64 upper profile features were extracted from the binary image of size 64-by-64.
- 64 horizontal histogram features and 64 vertical histogram features were extracted from the binary image of size 64-by-64.
- A feature vector of size 592 was labeled randomly, as the image class information was not known.
- This feature vector was sent to our holistic HWR classifier for recognition.

- Once the image was classified, LibSVM returned the results in probabilities. We assumed that this classified image belonged to a class that had the highest probability.
- The class with the highest probability did not mean that a word really belonged to it. To avoid these false positive errors, the image was passed to another phase called the Verification/Rejection phase. To avoid spotting the wrong word, we performed verification in order to reject or accept the image. This phase would make sure that the image actually belongs to a class on which our system was trained. If the image did not match with the class that had the highest probability then that image was rejected. The Verification/Rejection phase is explained further in section 5.4.3.

5.4.3 Verification/Rejection Phase

Once our system classified an image, we began the verification step in order to confirm that this classified image was actually one of the 57 words we had trained. This step was applied to avoid supplying false positives and to make sure that the classified word was one of the 57 words. Usually, our system gave a low probability for images that did not exist in our words database, but occasionally, a high confidence value was returned for these types of words. This led us to develop a rejection method in order to identify these images and to obtain a higher reliability rate for our system. Our system checked for the following rules in order to accept or reject a word image:

- High probability
- Number of dots in a word image

- Number of black pixels in a word image
- Height to width ratio of a word image
- Euclidean Distance Measurement

We will discuss each rule in detail in the following subsections.

5.4.3.1 High Probability

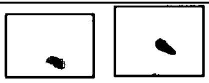
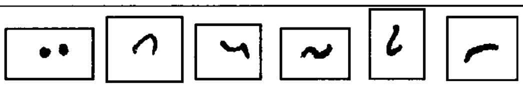
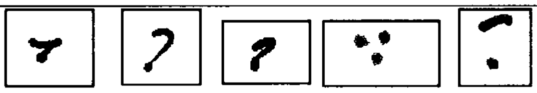
Our holistic HWR system provided the results in probabilities. We noticed that for our word spotting system, when the handwriting was clear, the HWR gave a very high probability for a word that belonged to a class that existed in our database. We put the threshold value higher than 0.95% to accept the word without any further acceptance rules. We also put the threshold value lower than 0.20% to reject the word without applying any further rejection rules.

5.4.3.2 Number of Dots

Like Farsi, 18 Urdu characters out of 39 are found with dots in a group of one, two or three. These dots can be found above or below the characters. In 2008, S. Mozaffari et al. [66] proposed a system for finding dots in handwritten Farsi/Arabic words for lexicon reduction (database reduction). Dots are the most commonly used diacritics and give important information about a word. As mentioned in section 5.3, some characters have the same main component shapes, but the number of dots and their positions distinguish those characters from each other.

As mentioned in section 5.3.2, in the diacritics recognition process, a single dot can be recognized easily. There are many variations in writing double dots and triple dots as shown in Table 5.7, below:

Table 5.7: Samples of Handwritten Single Dot, Double Dots and Triple Dots

	Printed	Handwritten samples
Single Dot		
Double Dots		
Triple Dots		

During the diacritics recognition process in word segmentation, we had already recorded how many dots were found in an image. For verification, our algorithm recognized the dots above and below the baseline in a word image. For each word class in CENPARMI's Urdu database, the dots were calculated from binary images by the following method:

- The skeleton of each image was taken using the Zhang-Suen algorithm.
- The baseline of a word image was estimated by a horizontal projection.
- Single dots were found easily. The skeleton of a single dot consists of one pixel or a maximum of four pixels.
- The diacritics recognition system was used to find the double as well as the triple dots.
- The total number of dots was calculated above and below the baseline.

- For each class, both the minimum and maximum numbers of dots were calculated.

Our verification method finds the total number of dots in each spotted word image. The number of dots in the image had to match with the number of dots in a word's respective class.

5.4.3.3 Number of Black Pixels

For each word class in our CENPARMI Urdu databases, we calculated the number of black pixels from the skeleton of each image. A 128-by-128 normalized image was used for this purpose. The minimum and maximum numbers of black pixels were calculated for each word class.

For each spotted word, the total number of black pixels from the skeleton was calculated. The value of this number was required to be between the minimum and maximum numbers of black pixels for all the word images in its respective word class.

5.4.3.4 Height to Width Ratio

For each word class in our CENPARMI Urdu databases, we calculated the height to width ratio of each 128-by-128 normalized image. For each word class, the height to width ratio of each image was calculated. From these calculated ratios, both the minimum and maximum ratios were found. The height and width of an Urdu word are shown in Fig 5.12, below:

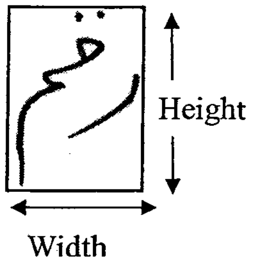


Fig. 5.12: Height and Width of an Urdu Word Image

For each spotted word, after 128-by-128 normalization, the height to width ratio was calculated. This ratio was expected to lie between the minimum and maximum ratios of all the word images in its respective class.

5.4.3.5 Euclidean Distance Measurement

The Euclidean distance finds the distance between two points in an *n*-space. If $p = (p_1, p_2, \ldots, p_n)$ and $q = (q_1, q_2, \ldots, q_n)$ are two points, then the Euclidean distance d between these two points is given by:

$$d = \sqrt{\sum_{i=1}^{n}(p_i - q_i)^2} \qquad (5.7)$$

For each word class in our CENPARMI Urdu database, we calculated its mean point by using the feature vectors. As mentioned in section 5.4.2.1, each feature vector had a size of 592 points. The distance of each sample in the word class was calculated by the Euclidean distance measurement. For each word class, the maximum distance was obtained from the mean.

The Euclidean distance between the spotted word image and the mean of its respective word class was calculated. If this distance was greater than the maximum distance of its respective word class, then the spotted word image was rejected. The calculation of the maximum distance of each class using the Euclidean distance measurement is given in the following formulas:

Let $V = (v_1, v_2, v_3, \ldots, v_{592})$ be the feature vector, then the mean value for each coordinate of $\overline{w}_i$ in each class is given by:

$$\overline{W}_i = \frac{\sum_{j=1}^{n} v_{ij}}{n} \tag{5.8}$$

where i = 1 to 592 (feature vector size) and n= number of samples in the class; in this case, 300 samples, and j = 1,2,3,....,n.

After calculating the mean value of each coordinate, the mean vector is obtained, which is given by:

$$\overline{w} = (\overline{w}_1, \overline{w}_2, \overline{w}_3, \ldots\ldots, \overline{w}_{592}) \tag{5.9}$$

After calculating the mean vector, the Euclidean distance between the feature vector and the mean of its class is calculated by:

$$\text{Distance } (\overline{w}, v_j) = \sqrt{\sum_{i=1}^{592} (\overline{w}_i - v_{ij})^2} \tag{5.10}$$

The maximum distance is also obtained for each class, shown below:

$$\text{Class max distance (d)} = \text{MAX } [\text{Distance}(\overline{w}, V_i)]_{i=1}^{300} \tag{5.11}$$

The maximum distance (d) for any class in a feature space is shown in Fig. 5.13, below:

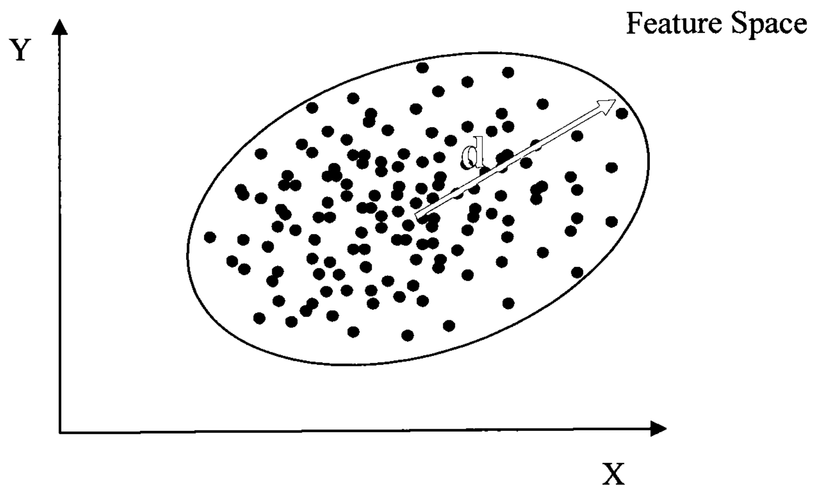


Fig. 5.13: Maximum Distance d From the Mean to the Outermost Point

Next, the Euclidean distance (p) between a spotted word and the mean of its class is calculated as follows:

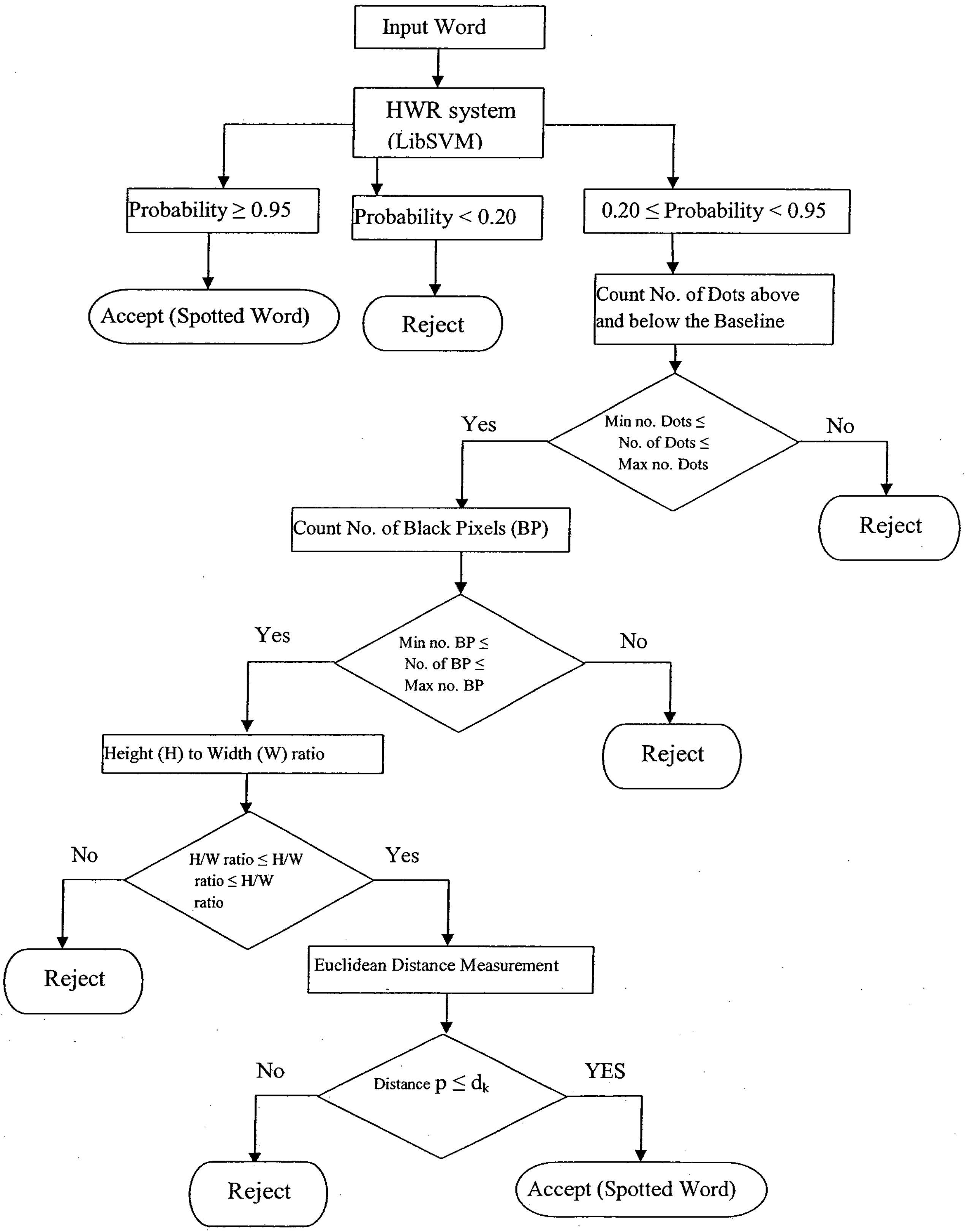
Let X = (x_1, x_2, x_3, ... , x_{592}) be the feature vector of a spotted word, where w_k is the mean of its class and d_k is the maximum distance of class *k*, which was calculated in Eqn. 5.10. The Euclidean distance between this feature vector and the mean of its class is given by:

$$p = \text{Distance}\ (\overline{w}_k, X) = \sqrt{\textstyle\sum_{i=1}^{592}(\overline{w}_i - x_i)^2}\,. \tag{5.12}$$

In order to accept the spotted word, the distance (P) should be less than or equal to the distance d_k:

To Accept: $(p \le d_k)$

The overall verification/rejection process is shown in Fig. 5.14, below:



Input Word
HWR system (LibSVM)
Probability ≥ 0.95
Probability < 0.20
0.20 ≤ Probability < 0.95
Accept (Spotted Word)
Reject
Count No. of Dots above and below the Baseline
Yes
Min no. Dots ≤ No. of Dots ≤ Max no. Dots
No
Reject
Count No. of Black Pixels (BP)
Yes
Min no. BP ≤ No. of BP ≤ Max no. BP
No
Reject
Height (H) to Width (W) ratio
No
H/W ratio ≤ H/W ratio ≤ H/W ratio
Yes
Reject
Euclidean Distance Measurement
No
Distance p ≤ dk
YES
Reject
Accept (Spotted Word)

Fig. 5.14: Verification/Rejection Processes Flowchart

Our Proposed Word Spotting architecture is shown in Fig. 5.15, below:

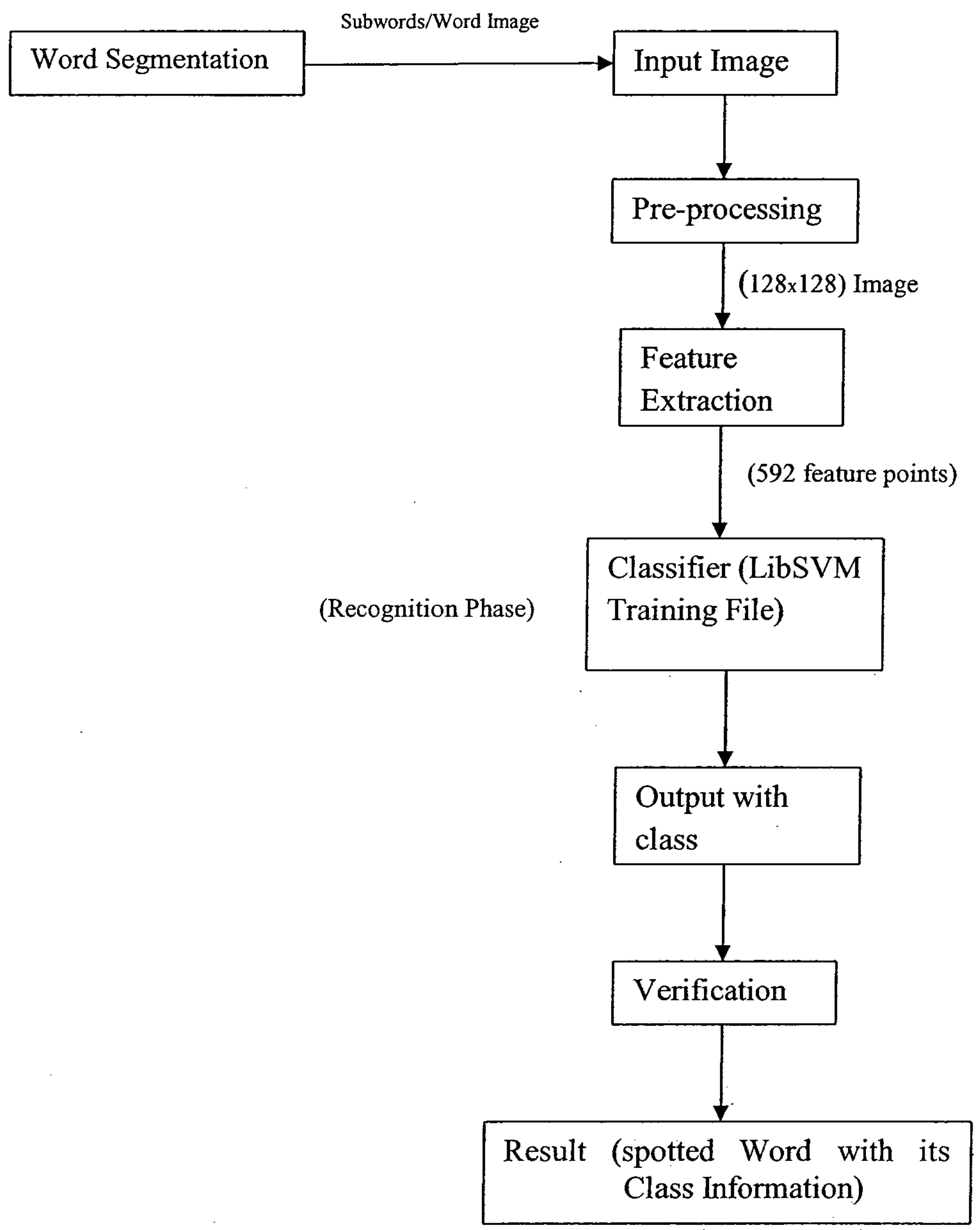


Fig. 5.15: Word Spotting System Flowchart

In the next subsection, we will discuss our experiments and results for word spotting.

5.4.4 Experiments and Results

In our word spotting experiments, we tested our system on each segmented text line written by ten writers. There were 57 words written by each writer in different locations on different text lines. These words were written a total of 86 times. Our system first found the candidate words on each handwritten text line. By using Eqs. 5.5 and 5.6, our word spotting system tested 3,000 (approximately) candidate words for each writer, as shown in Table 5.8. Overall, we achieved a 50% precision rate at a recall rate of 70%. Precision is the probability that a retrieved image is a target word and is defined by [70]:

$$\text{Precision} = \frac{\mathit{True\ Positive}}{\mathit{True\ Positive} + \mathit{False\ Positive}}$$

where *True Positive* (TP) is defined as the total number of correctly spotted target samples and *False Positive* (FP) is defined as the total number of spotted samples which were misrecognized.

Recall is defined by the successful retrieval of relevant target samples in the document and is given by:

$$\text{Recall} = \frac{\mathit{True\ Positive}}{\mathit{True\ Positive} + \mathit{False\ Negative}}$$

where *False Negative* (FN) is defined as the total number of target samples which were misspotted.

The spotting results for each writer and the overall results are shown in Table 5.8, below:

Table 5.8: Word Spotting Results

Writer No.	Total Candidate words	Correctly Rejected (TN)	Misrecognition (FP)	Misspotted (FN)	Correctly Spotted (TP)	%(Precision)
W0000	3062	2917	59	3	83	58.45
W0001	3132	3009	37	26	60	61.86
W0002	3260	3139	35	52	34	49.28
W0003	2972	2820	56	24	62	52.54
W0004	3244	3071	87	11	75	52.82
W0005	3154	3023	45	38	48	51.61
W0006	3230	3111	33	33	53	50.00
W0007	3408	3263	59	28	58	49.57
W0008	3284	3123	75	29	57	43.18
W0009	3118	2934	98	14	72	42.35
Overall	**31864**	**30410**	**584**	**258**	**602**	**50.76**

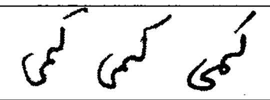
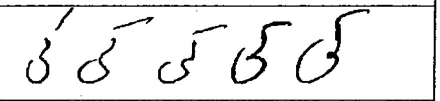
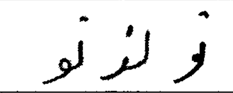
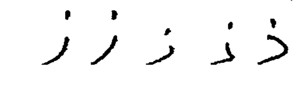
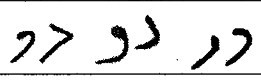
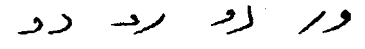
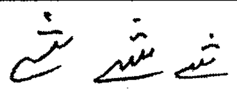
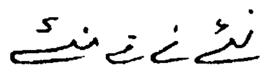
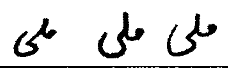
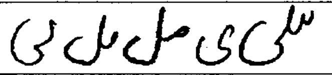
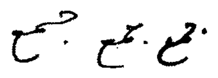
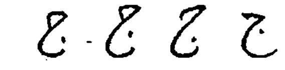
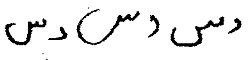
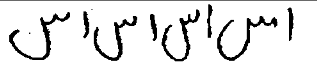
In the above table, column one shows the writer number and column two shows the total number of candidate words, which the system produced for each writer. Column three shows the total number of unseen samples which were rejected successfully also called *True Negatives* (TN). Column four shows *false positives*. Column five shows the number target samples which were misspotted, also called the *False Negative* (FN), and column six shows the number of correctly spotted words (*True Positives*). Column seven shows the precision percentage of correctly spotted words.

There were many unseen data (garbage data) and our system rejected them the majority of the cases. However, our system accepted words that did not exist in our database or accepted the spotted words that were very close to word shapes in our database. Also, there were some errors from the segmentation of the text lines. A detailed error analysis is presented in the next subsection.

5.4.4.1 Error Analysis in Word Spotting

Our word spotting system uses the holistic HWR system. As discussed in Chapter 4, section 4.3, our holistic HWR system extracted the directional and structural features. Because of the directional features, some types of errors in the spotting results came from words which were very similar to the word shapes in our databases and some errors came from the wrong segmentation of text lines. We found the seven errors which were repetitive for every writer. The seven most common types of errors are shown in Table 5.9, below:

Table 5.9: Word Spotting Error Analysis

Error No.	Word Name	Word Image	Handwritten Word Samples	Similar Spotted Images
1	Decrease	کمی		
2	Nine	نو		
3	Two	دو		
4	Item	شے		
5	Milli	ملی		
6	Debit	جمع		
7	Ten	دس		

In Table 5.9, the first column shows the error number, the second column shows the word name, the third column shows the word image, the fourth column shows the handwritten word samples taken from the training folder and the last column shows the wrongly spotted images.

In our experiments, the first error came from the shape (کُ), which is similar to the word shape decrease (کی) in the Urdu language. Error two came from the isolated letter Zey (ز), which is very similar to the word 'nine' shape in the Urdu language. In the Urdu language, the word 'two' (دو) is a combination of the two characters Dal (د) and Vao (و). In error three, the combination of the two isolated letters Re (ر) and Vao (و) is very similar to the word 'two' shape. This error came because of the similarity in isolated letters Re (ر) and Dal (د). This error is very confusing even for the human eye. The sixth error came from the isolated letter Jeem (ج) which is similar to the shape of a word 'debit' in the Urdu language. The frequency of the shape (کُ) is common in the Urdu language. The frequency of the errors that came from this word shape and the frequencies of other types of errors (from Table 5.9) are shown in Table 5.10, below:

Table 5.10: The Frequencies of Seven Errors (Listed in Table 5.9) for Each Writer

Error No	**Writer no.**									
	W0000	**W0001**	**W0002**	**W0003**	**W0004**	**W0005**	**W0006**	**W0007**	**W0008**	**W0009**
Error 1.	12	6	8	10	8	11	16	11	12	11
Error 2.	8	5	4	2	7	6	8	7	9	11
Error 3.	14	6	6	17	12	9	8	11	18	16
Error 4.	3	3	3	3	1	1	0	1	1	2
Error 5.	5	2	7	6	2	3	2	1	11	10
Error 6.	1	3	0	3	2	1	3	3	6	3
Error 7.	8	9	2	2	8	8	3	3	4	6
Total	51	34	30	43	40	39	40	37	61	59

The above table shows the frequencies of the seven error types, which were presented in Table 5.9. In the above table, the frequencies of these error types are presented for each writer. In our experiments, out of a total of 584 FP errors, 434 errors (75%) came from these seven types of errors and 150 errors (25%) came from unseen data.

In the next subsection, we will conclude our discussions on our word spotting system.

5.5 Conclusion

In this chapter, we discussed our word segmentation and word spotting systems. We also discussed the results and error analysis of our word spotting system. Overall, we achieved a 92.11% correct segmentation rate and a 50.75% word spotting precision rate. In word spotting, we found that most of the errors came from seven different types of errors. In the future, the spotting accuracy may be improved if we focus on these seven types of errors. Some of these errors could be eliminated either by analyzing the context of the Urdu language or if we perform character recognition in word spotting.

In Chapter 6, we will discuss the conclusion of this thesis and future work related to this research.

CHAPTER 6

Conclusions and Future Work

6.1 Conclusions

The goal of this thesis was to implement systems for offline Urdu handwritten word recognition and for word spotting. This was the first time that a system was designed for offline Urdu handwritten word recognition.

Our handwritten word recognition system consisted of holistic-based recognition such that the segmentation of the word into characters was not required. In this system, the basic pre-processing of images was performed. The gradient and structural features were extracted from word images in the feature extraction phase. This system extracted 592 feature sets and these features helped in enhancing the recognition results. This system was trained and tested on 57 words. Overall, we achieved a 97 % accuracy rate for handwritten word recognition by using the SVM classifier. We noticed that the results could be improved by adding more training data and more feature sets. This holistic HWR system was used for our word spotting system.

Our word spotting system consisted of two processes: the segmentation of handwritten Urdu text lines and the word spotting algorithm. We implemented the rule-based segmentation system for handwritten Urdu text lines. In this segmentation system, connected components were categorized into main connected components and diacritics. Each main connected component was grouped with its nearest diacritics and then these grouped components were segmented from the other main connected components in a text line. These segmented connected components were passed to the word spotting algorithm for recognition.

In the word spotting algorithm, the system first generated the candidate words from the segmented connected components. The candidate word could consist of one segmented connected component or a combination of up to four consecutive segmented connected components. These candidate words were sent to our holistic HWR system, which extracted the features and recognized the image as one of the 57 words, on which our holistic HWR system was trained. After classification, each image was sent to the verification/rejection phase. In this phase, the recognized image with a low probability (less than 0.20) was rejected and with a very high probability (greater then 0.95) was accepted as a correctly spotted word. For the remaining recognized image, the system checked the number of black pixels in the image, height to width ratio of the image, number of dots in the image and the Euclidean distance of the image from its class by using the 592 features. Our verification/rejection phase helped in rejecting the maximum number of unseen (garbage data) images. Overall, we achieved a 50% word spotting precision rate.

6.2 Future Work

We achieved satisfactory results for our holistic HWR system by using 57 classes. However, we intend to increase the number of classes by collecting more handwritten data in the future. We also intend to collect more samples for these 57 classes.

For our word spotting system, results may be improved for the word segmentation if we perform recognition-based segmentation. The correct segmentation of words could improve the word spotting results.

There are some unique characteristics for the writing of Urdu text. One of the important characteristics is: There are 29 isolated letters which never appear at the beginning of a word and if they appear in any position on a text line, then that means the word is ending there. If we train our system on those isolated letters, then it will help in generating the candidate words and it will also prevent the system from generating the garbage data. Those 29 Urdu isolated letters, which do not appear at the start of the words, are shown in the Table 6.1, below:

Table 6.1: The Urdu Isolated Letters Which Do Not Appear at the Beginning of a Word in Isolated Form

س	خ	ح	چ	ج	ث	ٹ	ت	پ	ب
Seen	**Khe**	**Hey**	**Che**	**Jim**	**Se**	**Tay**	**Te**	**pe**	**Be**
ک	ق	ف	غ	ع	ظ	ط	ض	ص	ش
Kaf	**Qaf**	**Fey**	**Gain**	**Ain**	**Zoe**	**Toe**	**Zvad**	**Swad**	**Sheen**
	ی	ے	ء	ھ	ہ	ن	م	ل	گ
	Bardi Yey	**Choti Yey**	**Hamza**	**He dow chashm**	**choti Hay**	**Noon**	**Mim**	**Lam**	**Gaf**

In our system for segmentation of text line, we found that most of the errors came from the isolated letter alif (ا), which was wrongly segmented, and from a diacritic, the single line for kaaf (/). The segmentation results could also be improved if we would correctly recognize diacritics and the isolated letter Alif (ا) in a text line. For diacritic recognition, we have already created a database for the diacritics. From our CENPARMI Urdu databases, we extracted the diacritics and made some new diacritics databases. We divided these databases into three different sets - Training, Testing and Validation. The sizes of the training set and the testing set are shown in Table 6.2, below:

Table 6.2: The Number of Diacritics in the Training and Testing Sets

Diacritic name	Training set	Testing set
Double Dot	1012	339
Hamza	228	76
Hey	224	82
Mudd	127	43
Pesh	86	29
Small Toway	967	323
Single line for kaaf	1172	391
Double lines for Gaaf	103	34

The word spotting results could also be improved by working on the seven most common errors, which are mentioned in chapter five, section 5.4.4.1. We noticed that 63% of the errors came from those seven types of errors. Those errors could be minimized if we train our system for recognizing each type of error. For example, error one, we may implement a binary classifier to train our system on two classes, which include the word decrease (کمی) and its similar shape (کی), and we may also implement a binary classifier for each error. This would help in minimizing the errors in word spotting.

References

[1] S. A. Sattar, S. Haque, M. K. Pathan and Q. Gee, "Implementation Challenges for Nastaliq Character Recognition," *in the Proc. of the Wireless Networks, Information Processing and Systems International Multi Topic Conference*, IMTIC 2008 Jamshoro, Pakistan, pp. 279-285, April 11-12, 2008.

[2] S. Malik and S. A. Khan, "Urdu Online Handwriting Recognition," *in the Proc. of the International Conference on Emerging Technologies,* IEEE 2005, pp. 27-31, Islamabad, Pakistan, Sept. 17-18, 2005.

[3] M. I. Razzak, S. A. Hussain, M. Sher and Z. S. Khan, "Combining Offline and Online Preprocessing for Online Urdu Character Recognition," *in the Proceedings of the International MultiConference of Engineers and Computer Scientists (IMECS 2009)*, vol. I, pp. 912-915, Hong Kong, March 18 - 20, 2009.

[4] A. Benouareth, A. Ennaji and M. Sellami, "Arabic Handwritten Word Recognition Using HMMs with Explicit State Duration," *EURASIP Journal on Advances in Signal Processing*, vol. 2008, Article ID 247354, pp. 1-13, 2008.

[5] L. M. Lorigo and V. Govindaraju, "Offline Arabic Handwriting Recognition: A Survey," *IEEE transactions on Pattern Analysis and Machine Intelligence*, vol. 28 no. 5, pp. 712-724, May 2006.

[6] H. Al-Rashaideh, "Preprocessing phase for Arabic word handwritten Recognition," *Russian Academy of Sciences*. 6(1), pp. 11-19. Russian Federation. 2006.

[7] F. Farooq, V. Govindaraju and M. Perrone, "Pre-processing Methods for Handwritten Arabic Documents," *in the proceedings of the Int'l conf. Document Analysis and Recognition (ICDAR)*, vol. 1, pp. 267-271, Seoul, Korea, September 2005.

[8] J. AlKhateeb, J. Ren, S. S. Ipson and J. Jiang, "Knowledge-based Baseline Detection and Optimal Thresholding for Words Segmentation in Efficient Pre-processing of Handwritten Arabic Text," *in the Proc. of the Fifth International Conference on Information Technology: New Generations (ITNG)*, 2008, pp.1158-1159, Las Vegas, Nevada, USA, April 2008.

[9] M. Dehghan, K. Faez, M. Ahmadi and M. Shridhar, "Handwritten Farsi (Arabic) word recognition: a holistic approach using discrete HMM," *In: Pattern Recognition*, vol. 34, no. 5, pp. 1057 – 1065, 2001.

[10] A. M. Al-Shatnawi and K. Omar, "Methods of Arabic Language Baseline Detection – The State of Art," International Journal of Computer Science and Network Security (IJCSNS), vol. 8, no. 10, pp. 137-143, Oct. 2008.

[11] M. Pechwitz and V. Maergner, "Baseline estimation for Arabic handwritten words," *In the proceedings of the Eighth International Workshop on Frontiers in Handwriting Recognition (IWFHR)*, pp. 479–484, Niagara-on-the-Lake, Ontario, Canada, Aug. 2002.

[12] A. Amin, H. Al-Sadoun and S. Fischer, "Hand-Printed Arabic Character Recognition System Using an Artificial Network," *In: Pattern Recognition*, vol. 29, pp. 663-675, 1996.

[13] I.S.I. Abuhaiba, M.J.J. Holt and S. Datta, "Recognition of Off-Line Cursive Handwriting," *Computer Vision and Image Understanding*, vol. 71, No. 1, pp. 19-38, 1998.

[14] S. Mozaffari, K. Faez and M. Ziaratban, "Structural Decomposition and Statistical Description of Farsi/Arabic Handwritten Numeric Characters," *in the proceedings of the Int'l Conf. Document Analysis and Recognition (ICDAR)*, vol. 1, pp. 237-241, Seoul, Korea, September 2005.

[15] T.Y. Zhang and C.Y. Suen, "A fast parallel algorithm for thinning digital patterns," *Communications of the ACM*, vol. 27, No. 3, pp. 236-239, March 1984.

[16] H. Almuallim and S. Yamaguchi, "A Method of Recognition of Arabic Cursive Handwriting," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 9, pp. 715-722, 1987.

[17] T. Sari, L. Souici and M. Sellami, "Off-Line Handwritten Arabic Character Segmentation Algorithm: ACSA," *In the proceedings of the Eighth International Workshop on Frontiers in Handwriting Recognition (IWFHR)*, pp. 452-457, Niagara-on-the-Lake, Ontario, Canada, Aug. 2002.

[18] S. Snoussi Maddouri, H. Amiri, A. Belaid and C. Choisy, "Combination of Local and Global Vision Modeling for Arabic Handwritten Words Recognition," *In the proceedings of the Eighth International Workshop on Frontiers in Handwriting Recognition (IWFHR)*, pp. 128-135, Niagara-on-the-Lake, Ontario, Canada, Aug. 2002.

[19] M. Dehghan, K. Faez, M. Ahmadi and M. Shridhar, "Handwritten Farsi (Arabic) Word Recognition: A Holistic Approach Using Discrete HMM," *In: Pattern Recognition*, vol. 34, pp. 1057-1065, 2001.

[20] K. Mostafa and A.M. Darwish, "Robust Base-Line Independent Algorithms for Segmentation and Reconstruction of Arabic Handwritten Cursive Script," *In the proceedings of the IS&T/SPIE Conf. Document Recognition and Retrieval VI*, vol. 3651, pp. 73-83, San Jose, CA, January 1999.

[21] R. Safabakhsh and P. Adibi, "Nastaaligh Handwritten Word Recognition Using a Continuous-Density Variable-Duration HMM," *The Arabian Journal for Science and Engineering,* vol. 30, pp. 95-118, 2005.

[22] M.-Y. Chen, A. Kundu and S.N. Srihari, "Variable Duration Hidden Markov Model and Morphological Segmentation for Handwritten Word Recognition," *IEEE Trans. Image Processing*, vol. 4, pp. 1675-1688, 1995.

[23] R. El-Hajj, L. Likforman-Sulem and C. Mokbel, "Arabic Handwriting Recognition Using Baseline Dependant Features and Hidden Markov Modeling," *In the proceedings of the Int'l Conf. Document Analysis and Recognition (ICDAR)*, pp. 893-897, Seoul, Korea, September 2005.

[24] M.S. Khorsheed, "Recognising Handwritten Arabic Manuscripts Using a Single Hidden Markov Model," *Pattern Recognition Letters*, vol. 24, pp. 2235-2242, 2003.

[25] M. Pechwitz and V. Maergner, "HMM Based Approach for Handwritten Arabic Word Recognition Using the IFN/ENIT-Database," *In the proceedings of the Int'l Conf. Document Analysis and Recognition (ICDAR)*, pp. 890-894, Edinburgh, Scotland, UK August 2003.

[26] L.R. Rabiner, "A tutorial on hidden Markov models and selected applications in speech recognition," *In the proceedings of the IEEE,* vol. 77 (2), pp. 257-286, 1989.

[27] G. R. Ball, S. N. Srihari and H. Srinivasan, "Segmentation-Based And Segmentation-Free Methods for Spotting Handwritten Arabic Words," *Tenth International Workshop on Frontiers in Handwriting Recognition*, La Baule, France, October 23-26, 2006.

[28] S. N. Srihari, H. Srinivasan, P. Babu and C. Bhole, "Handwritten Arabic Word Spotting using the CEDARABIC Document Analysis System," *In Proc. Symposium on Document Image Understanding Technology (SDIUT-05)*, pp. 123–132, Adelphi, Maryland, USA, November, 2005.

[29] S. N. Srihari, H. Srinivasan, P. Babu and C. Bhole, "Spotting Words in Handwritten Arabic Documents," *In Document Recognition and Retrieval XIII: Proceedings SPIE,* San Jose, CA, pp. 606702-1–606702-12, 2006.

[30] R. Saabni and J. El-Sana, "Keyword Searching for Arabic Handwritten Documents," *In the proceedings of the International Conference on Frontiers in Handwriting Recognition (ICFHR)*, pp 271-277, Montreal, Canada, Aug 19-21, 2008.

[31] T. Rath and R. Manmatha, "Word image matching using dynamic time warping," *Computer Vision and Pattern Recognition, In the Proceedings of the IEEE Computer Society Conference*, vol. 2, pp. II–521–II–527, Madison, WI, USA, 18-20 June 2003.

[32] S. N. Srihari, B. Zhang, C. Tomai, S. Lee and Y. C. Shin, “A system for handwriting matching and recognition,” *In the Proceedings of the 2003 Symposium on Document Image Understanding Technology (SDIUT03)*, pp. 67-75 Greenbelt, MD, USA, April 2003.

[33] M. Egmont-Petersen, D. de Ridder and H. Handels, “Image processing with neural networks - a review,” *Pattern Recognition*, vol. 35, Issue 10, pp. 2279-2301, October 2002.

[34] W. Anwar, X. Wang, and X.-L Wang, “A survey of automatic Urdu language processing,” *In the Proceedings of the Fifth International Conference on Machine Learning and Cybernetics*, pp. 13–16, Dalian, China, 2006.

[35] C.-L. Liu, K. Nakashima, H. Sako and H. Fujisawa, “Handwritten digit recognition, Investigation of normalization and feature extraction techniques,” *In: Pattern Recognition*, vol. 37, no. 2, pp. 265-279, 2004.

[34] C.-L. Liu, and C. Y. Suen, “A new benchmark on the recognition of handwritten Bangla and Farsi numeral characters,” *In the Proceedings of the 11th International Conference on Frontiers in Handwriting Recognition (ICFHR)*, pp. 278 - 283. Montreal, Canada, Aug 19-21, 2008.

[35] N. Otsu, “A threshold selection method from gray-level histogram,” *In: IEEE Trans. System Man Cybernetics*, vol. 9, pp. 1569-1576, 1979.

[36] M. Shi, Y. Fujisawa, T. Wakabayashi and F. Kimura, “Handwritten numeral recognition using gradient and curvature of gray scale image,” *In Pattern Recognition*, vol. 35, no. 10, pp. 2051-2059, 2002.

[37] M. Pechwitz, S. Snoussi Maddouri, V. Märgner, N. Ellouze and H. Amiri, “IFN/ENIT-Database of handwritten Arabic words,” *In the 7th Colloque International Francophone sur l'Ecrit et le Document (CIFED)*, pp. 129–136, Hammamet, Tunis, Oct. 21-23, 2002.

[38] H. Alamri, J. Sadri, N. Nobile, and C. Y. Suen, “A Novel Comprehensive Database for Arabic Off-Line Handwriting Recognition,” *In the proceedings of the 11th International Conference on Frontiers in Handwriting Recognition (ICFHR)*, pp. 664–669, Montreal, Canada, 2008.

[39] M. I. Shah, J. Sadri, N. Nobile, and C. Y. Suen, “A New Multipurpose Comprehensive Database for Handwritten Dari recognition,” *In the proceedings of the 11th International Conference on Frontiers in Handwriting Recognition (ICFHR)*, pp. 635–640, Montreal, Canada, Aug 19-21, 2008.

[40] P. Jifroodian Haghighi, N. Nobile, C. L. He and C. Y. Suen, “A New Large-Scale Multi-Purpose Handwritten Farsi Database,” *In the Proceedings of the International*

Conference on Image Analysis and Recognition (ICIAR), pp. 278-286. Halifax, July 2009.

[41] M. I. Shah, C. L. He, N. Nobile and C. Y. Suen, "A Handwritten Pashto Database With Multi-Aspects for Handwriting Recognition," *In the Proceedings of the 14th Conference of the International Graphonomics Society*, pp. 157-161, Dijon, France, Sept. 2009.

[42] Y. Al-Ohali, M. Cheriet, and C. Y. Suen, "Databases for recognition of handwritten Arabic cheques," *Pattern Recognition*, vol. 36, pp. 111-121, 2003.

[43] S. Mozaffari, H.E. Abed, V. Margner, K. Faez and A. Amirshahi, "IFN/Farsi-Database: A Database of Farsi Handwritten City Name," *In the Proceedings of the 11th International Conference on Frontiers in Handwriting Recognition (ICFHR)*, pp. 397-402, Montreal, Canada, Aug 19-21, 2008.

[44] A.M. Bidgoli and M. Sarhadi, "IAUT/PHCN: Azad University of Tehran / Persian Handwritten City Names, a very large database of handwritten Persian word," *In the proceedings of the 11th International Conference on Frontiers in Handwriting Recognition (ICFHR)*, pp. 192-197, Montreal, Canada, Aug 19-21, 2008.

[45] M. W. Sagheer, C. L. He, N. Nobile and C. Y. Suen, "A New Large Urdu Database for Off-Line Handwriting Recognition," *In the Proceedings of the International Conference on Image Analysis and Processing (ICIAP),* Salerno, Italy, pp. 538-546, Sept 8-11, 2009.

[46] Urdu, *A Language of Pakistan*, retrieved October 18, 2009, from http://www.ethnologue.com/show_language.asp?code=urd

[47] C.-L. Liu, K. Nakashima, H. Sako and H. Fujisawa, "Handwritten digit recognition, Investigation of normalization and feature extraction techniques," *Pattern Recognition*, vol. 37, no. 2, pp. 265-279, 2004.

[48] Z. Aghbari and S. Brook, "HAH manuscripts: A holistic paradigm for classifying and retrieving historical Arabic handwritten documents," *Expert Systems with Applications: An International Journal*, vol. 36, No. 8, pp. 10942-10951, 2009.

[49] J.A. Rodríguez and F. Perronnin, "Local gradient histogram features for word spotting in unconstrained handwritten documents," *In the proceedings of the 11th International Conference on Frontiers in Handwriting Recognition (ICFHR)*, pp. 7–11, Montreal, Canada, Aug 19-21, 2008.

[50] J.A. Rodríguez and F. Perronnin, "Handwritten word-spotting using hidden Markov models and universal vocabularies," *Pattern Recognition*, vol. 42, pp. 2106-2116, September 2009.

[51] L.G. Roberts, "Machine Perception of Three-Dimensional Solids," *Optical Electro-Optical Processing of Information*, MIT Press, pp. 159–197, Cambridge, Massachusetts, 1965.

[52] J.C. Russ, "The Image Processing Handbook," *2nd Edition*, CRC Press, Boca Raton, USA, 1995.

[53] B. Gatos, I. Pratikakis, A. L. Kesidis and S. J. Perantonis, "Efficient Off-Line Cursive Handwriting Word Recognition," *In the Proceedings of the Tenth International Workshop on Frontiers in Handwriting Recognition (IWFHR)*, inria-00104302, ver. 1 La Baule, October 23-26, 2006.

[54] F. Camastra, M. Spinetti and A. Vinciarelli, "Offline Cursive Character Challenge: a New Benchmark for Machine Learning and Pattern Recognition Algorithms," *18th International Conference on Pattern Recognition (ICPR'06)*, vol. 2, pp.913-916, Hong Kong, August 2006.

[55] R. Hamdi, F. Bouchareb and M. Bedda, "Handwritten Arabic character recognition based on SVM Classifier," *3rd International Conference on Information and Communication Technologies: From Theory to Applications (ICTTA)*, pp.1-4, Damascus, Syria, 7-11 April 2008

[56] C.-C. Chang and C.-J. Lin, *LIBSVM -- A Library for Support Vector Machines.*

[57] A. R. Ahmad, C. Viard-Gaudin and M. Khalid, "Lexicon-Based Word Recognition Using Support Vector Machine and Hidden Markov Model," *10th International Conference on Document Analysis and Recognition (ICDAR)*, pp. 161-165, Barcelona, Spain, July 2009

[58] *Support Vector Machines*, retrieved September 21, 2009, from http://www.statemaster.com/encyclopedia/Support-vector-machines

[59] C.J.C. Burges, "A tutorial on support vector machines for pattern recognition." *Data Mining and Knowledge Discovery*, vol. 2, pp. 121-167, 1998.

[60] *Kernel Machines*, retrieved September 21, 2009, from http://www.kernel-machines.org/

[61] S. Abe, "Support Vector Machines for Pattern Classification." *Springer-Verlag, London*, 2005.

[62] C. Cortes and V. Vapnik, "Support-Vector Networks," *Machine Learning*, vol. 20, no. 3, pp. 273-297, 1995.

[63] A. Gidudu, G. Hulley and T. Marwala, "Image classification using svms: One-againstone vs one-against-all," *In the Proceedings of the 28th Asian Conference on Remote Sensing (ACRS)*, Article No. 0711.2914, Kuala Lumpur, Malaysia, 2007

[64] T. Y. Zhang, C. Y. Suen, "A fast parallel algorithm for thinning digital patterns," *Communications of the ACM*, vol. 27, no. 3, pp. 236-239, March 1984.

[65] S. N. Srihari, G. R. Ball and H. Srinivasan, "Versatile Search of Scanned Arabic Handwriting," *In the Proceedings of the Summit on Arabic and Chinese Handwriting (SACH)*, College Park, MD, USA, pp. 151-164, September 2006.

[66] S. Mozaffari, K. Faez, V. Märgner and H. El-Abed, "Lexicon reduction using dots for off-line Farsi/Arabic handwritten word recognition." *Pattern Recognition Letter*, vol. 29, Issue 6, pp. 724-734, Apr. 2008.

[67] M. Cheriet, N. Kharma, C.-L. Liu and C. Y. Suen, "Character recognition systems: a guide for students and practitioners," *Wiley*, 2007.

[68] R. Plamondon and S. Srihari, "On-line & Off-line Handwriting Recognition: A Comprehensive Survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no.1, pp. 63-84, 2000.

[69] P. Jifroodian Haghighi, "A Discrete HMM for Recognition of Handwritten Frasi Words," *Master's Thesis, Concordia University,* January 12, 2010.

[70] D. L. Olson, D. Delen, "Advanced Data Mining Techniques," *Springer*; 1 edition, page 138, February 1, 2008.

Appendix A: The Complete data sets and distribution of data in each class

Databases		Total	Training	Testing	Validation
Dates		278	166	56	56
Isolated_Characters	**Aday**	343	206	69	68
	Alif	343	206	69	68
	Alif Mud aa	343	206	69	68
	Bardi ye	343	206	69	68
	Bardi ye hamza	343	206	69	68
	Bey	343	206	69	68
	Chey	343	206	69	68
	Chooti ye	343	206	69	68
	Choti ye hamza	343	206	69	68
	Dal	343	206	69	68
	Ein	343	206	69	68
	Fay	343	206	69	68
	Gaaf	343	206	69	68
	Gein	343	206	69	68
	Hamza	343	206	69	68
	Hey	343	206	69	68
	Hey Hamza	343	206	69	68

Isolated_Characters	Hiy	343	206	69	68
	Hiy(do chashm)	343	206	69	68
	Jeem	343	206	69	68
	Kaaf	343	206	69	68
	Khey	343	206	69	68
	Laam	343	206	69	68
	Meem	343	206	69	68
	Noon	343	206	69	68
	Noon Guna	343	206	69	68
	Pey	343	206	69	68
	Qaaf	343	206	69	68
	Rey	343	206	69	68
	Seen	343	206	69	68
	Sey	343	206	69	68
	Sheen	343	206	69	68
	Sowad	343	206	69	68
	Tay	343	206	69	68
	Thahy	343	206	69	68
	The al	343	206	69	68
	Toway	343	206	69	68
	Ty	343	206	69	68
	Wao	343	206	69	68
	Zaal	343	206	69	68

		Zay	343	206	69	68
		Zhay	343	206	69	68
		Zowad	343	206	69	68
		Zoway	343	206	69	68
Isolated_Digits		Isolated_0	3602	2161	720	721
		Isolated _1	3072	1843	614	615
		Isolated _2	3186	1910	638	638
		Isolated _3	3314	1988	663	663
		Isolated _4	2958	1774	592	592
		Isolated _5	2833	1699	566	568
		Isolated _6	3019	1811	604	604
		Isolated _7	3077	1845	616	616
		Isolated _8	2701	1621	540	540
		Isolated _9	3424	2054	685	685
Numeral Strings	**Integer_strings**	Lenght_2	3807	2283	761	763
		Lenght_3	2049	1227	411	411
		Lenght_4	1266	758	254	254
		Lenght_6	1465	878	293	294
		Lenght_7	1465	878	293	294
	Real_strings(with_ Decimal_Points)	Length_4	586	350	118	118
		Length_5	343	206	69	68
		Isolated_ Dot	529	317	106	106
		at_the_rate	343	206	69	68

Special_Symbols	**Colon**	343	206	69	68
	Comma	343	206	69	68
	Forward Slash	343	206	69	68
	Hash	343	206	69	68
Words	**Amount**	343	206	69	68
	Article	343	206	69	68
	Balance	343	206	69	68
	Carton	343	206	69	68
	Cash	343	206	69	68
	Centi	343	206	69	68
	Cost	343	206	69	68
	Credit	343	206	69	68
	Debit	343	206	69	68
	Decrease	343	206	69	68
	Delivery	343	206	69	68
	Dozen	343	206	69	68
	Due	343	206	69	68
	Duration	343	206	69	68
	Duty	343	206	69	68
	Eight	343	206	69	68
	Expire	343	206	69	68
	Five	343	206	69	68
	Four	343	206	69	68

Words	Gallon	343	206	69	68
	Gram	343	206	69	68
	Hundred	343	206	69	68
	Inch	343	206	69	68
	Increase	343	206	69	68
	Interest	343	206	69	68
	Inventory	343	206	69	68
	Item	343	206	69	68
	Kilo	343	206	69	68
	Lease	343	206	69	68
	Length	343	206	69	68
	Liter	343	206	69	68
	Meter	343	206	69	68
	Mili	343	206	69	68
	Nine	343	206	69	68
	No.	343	206	69	68
	One	343	206	69	68
	Part	343	206	69	68
	Payment	343	206	69	68
	Plus	343	206	69	68
	Price	343	206	69	68
	Product	343	206	69	68
	Rent	343	206	69	68
	Rupees	343	206	69	68

	Seven	343	206	69	68
	Stock	343	206	69	68
	Tax	343	206	69	68
	Ten	343	206	69	68
	Thousand	343	206	69	68
	Three	343	206	69	68
	Ton	343	206	69	68
	Total	343	206	69	68
	Transfer	343	206	69	68
	Two	343	206	69	68
	Volume	343	206	69	68
	Weight	343	206	69	68
	Width	343	206	69	68

Appendix B: Confusion Matrix

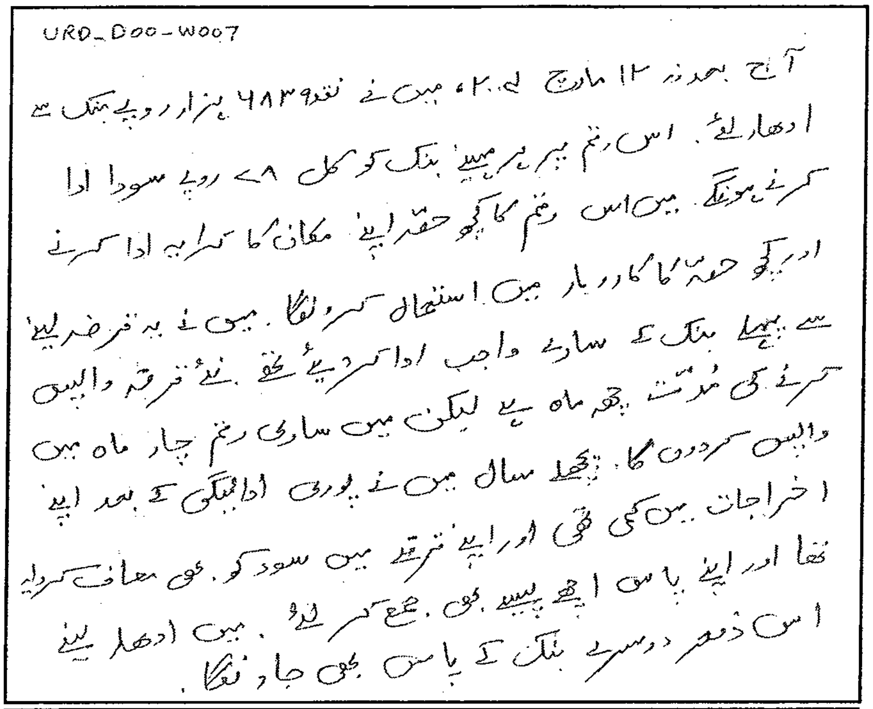
		Output																													
		01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
True Label	01	67																															
	02		64						2																								
	03			66																												1	
	04				62													3						1									
	05					61																											
	06						65	1		2																							
	07							64																			1						
	08								67																								
	09									64									1														
	10				2						61							1				1		2			1						
	11											63										1			1								
	12						1						65								1												
	13													68																		1	
	14														64					1													
	15															61																	
	16																69																
	17		1												1			63															
	18																		65													2	
	19																			65		1											
	20																				65												
	21																					64									2		1
	22											1											63										
	23		1											2										59									
	24				1																				64								
	25																									64							
	26																										63						
	27															1											1	65			1		
	28																											1	62	1			
	29											2	1																1	64			
	30									1												2								1	60		
	31												2																		1	62	
	32																			1		1										1	59
	33																		1														1
	34																																
	35				1	1					1																						
	36																																
	37																				2												
	38									2																							
	39																																
	40																								1								
	41														1									1									
	42																								1								
	43																								1								
	44																							1									
	45				1																			1									
	46																																
	47																												1				
	48																																
	49					1					1		1																				
	50		1																						1								
	51	1				2																											
	52																		1														
	53																		1			1											
	54		2										1																				
	55																											1					
	56																																
Reliability	Cor.	67	64	66	62	61	65	64	67	64	61	63	65	68	64	61	69	63	65	65	65	64	63	59	64	64	63	65	62	64	60	62	59
	Inc.	1	5	0	5	4	1	1	2	5	2	3	5	2	2	1	0	4	4	2	3	7	0	6	5	0	3	2	2	2	4	5	2
	Total	68	69	66	67	65	66	65	69	69	63	66	70	70	66	62	69	67	69	67	68	71	63	65	69	64	66	67	64	66	64	67	61
	Pct.	98.5%	92.8%	100.0%	92.5%	93.8%	98.5%	98.5%	97.1%	92.8%	96.8%	95.5%	92.9%	97.1%	97.0%	98.4%	100.0%	94.0%	94.2%	97.0%	95.6%	90.1%	100.0%	90.8%	92.8%	100.0%	95.5%	97.0%	96.9%	97.0%	93.8%	92.5%	96.7%
	Rank	7	43	1	46	38	12	13	16	43	25	29	41	15	21	14	1	35	32	19	28	52	1	50	43	1	29	19	24	21	40	46	26

Appendix B: (a) Confusion Matrix HWR Results for classes 1 to 32.

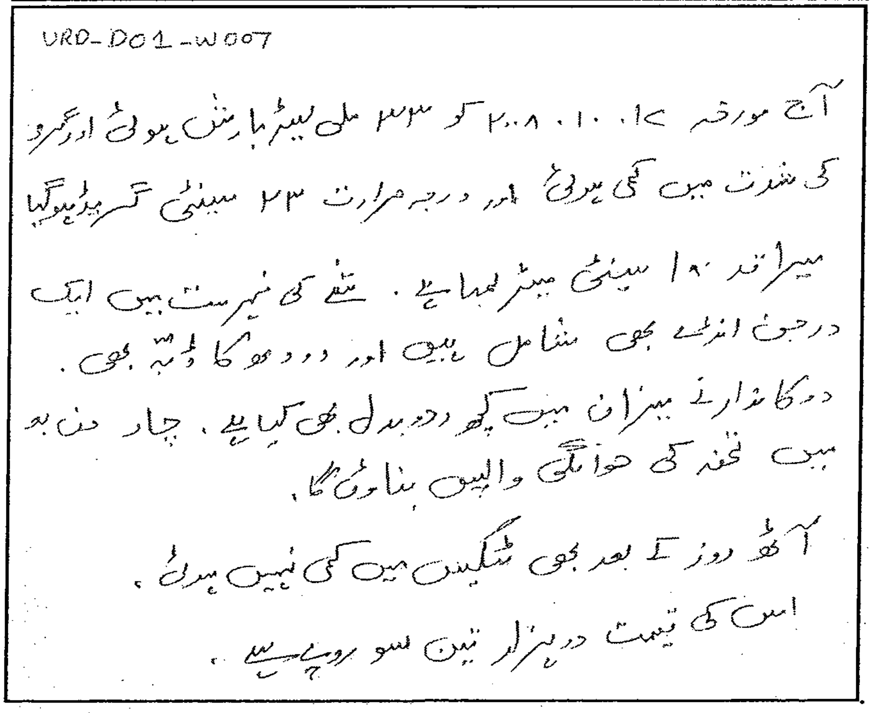
																										Accuracy				
		33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	Cor.	Inc.	Total	Pct.	Rank
True Label	01			1																						67	1	68	98.5%	6
	02													1												64	3	67	95.5%	31
	03																					1				66	2	68	97.1%	15
	04																1									62	5	67	92.5%	46
	05		1	1																			2			61	4	65	93.8%	43
	06																									65	3	68	95.6%	27
	07																1									64	2	66	97.0%	20
	08										1															67	1	68	98.5%	6
	09								1																	64	2	66	97.0%	20
	10										1															61	8	69	88.4%	53
	11		1																				1			63	4	67	94.0%	40
	12																				1					65	3	68	95.6%	27
	13																									68	1	69	98.6%	5
	14																		1			1				64	3	67	95.5%	31
	15							1												1				1		61	3	64	95.3%	35
	16																									69	0	69	100.0%	1
	17																									63	2	65	96.9%	22
	18																									65	2	67	97.0%	17
	19	1																	1			1				65	4	69	94.2%	36
	20																									65	0	65	100.0%	1
	21																						1			64	4	68	94.1%	38
	22																1						1			63	3	66	95.5%	34
	23									1				1												59	5	64	92.2%	51
	24								1							1										64	3	67	95.5%	31
	25																									64	0	64	100.0%	1
	26																	1								63	1	64	98.4%	10
	27																									65	3	68	95.6%	27
	28		1						1							3										62	7	69	89.9%	52
	29																									64	4	68	94.1%	38
	30		1																							60	5	65	92.3%	50
	31						1																			62	4	66	93.9%	41
	32													1												59	4	63	93.7%	44
	33	65	1																1							65	4	69	94.2%	36
	34	1	66																			1				66	2	68	97.1%	15
	35		1	61														1								61	5	66	92.4%	48
	36			1	67		1																			67	2	69	97.1%	11
	37	2			1	61																				61	5	66	92.4%	48
	38						67																			67	2	69	97.1%	11
	39						1	67																		67	1	68	98.5%	6
	40							1	65																	65	2	67	97.0%	17
	41								1	60	4											1				60	8	68	88.2%	54
	42									4	60			1								2				60	8	68	88.2%	54
	43										1	66											1			66	3	69	95.7%	24
	44											1	67													67	2	69	97.1%	11
	45												1	65												65	3	68	95.6%	27
	46													1	67											67	1	68	98.5%	6
	47														1	65										65	2	67	97.0%	17
	48													1		1	66						1			66	3	69	95.7%	24
	49			1													1	62								62	5	67	92.5%	46
	50		1															1	62							62	4	66	93.9%	41
	51																1		1	63						63	5	68	92.6%	45
	52																			1	67					67	2	69	97.1%	11
	53																				1	66				66	3	69	95.7%	24
	54		1														2					1	60	2		60	9	69	87.0%	56
	55																	1					1	67		67	3	70	95.7%	23
	56																							1	69	69	1	70	98.6%	4
Reliability	Cor.	65	66	61	67	61	67	67	65	60	60	66	67	65	67	65	66	62	62	63	67	66	60	67	69	3589	181	3770	95.2%	
	Inc.	4	8	4	1	0	3	2	4	5	7	1	1	6	1	5	7	4	4	2	2	8	8	4	0					
	Total	69	74	65	68	61	70	69	69	65	67	67	68	71	68	70	73	66	66	65	69	74	68	71	69					
	Pct.	94.2%	89.2%	93.8%	98.5%	100.0%	95.7%	97.1%	94.2%	92.3%	89.6%	98.5%	98.5%	91.5%	98.5%	92.9%	90.4%	93.9%	93.9%	96.9%	97.1%	89.2%	88.2%	94.4%	100.0%					
	Rank	32	54	38	7	1	27	16	32	48	53	11	7	49	7	41	51	36	36	23	16	54	56	31	1					

Appendix B: (b) Confusion Matrix HWR Results for classes 33 to 56

Appendix C: Samples of Handwritten Text Pages for writer #W0007

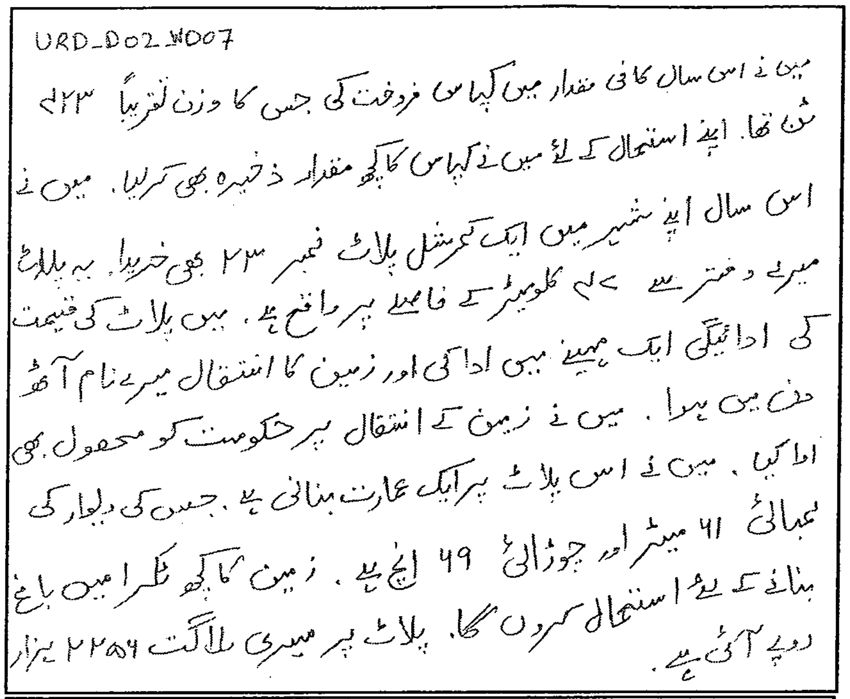

URD_D00-W007

آج بمورخہ ۱۲ مارچ ۲۰۰۷ء میں نے نقد ۴۸۳۹ ہزار روپے بنک سے
ادھار لئے۔ اس رقم پر ہر مہینے بنک کو کل ۸۷ روپے سود ادا
کرنے ہونگے۔ میں اس رقم کا کچھ حصہ اپنے مکان کا کرایہ ادا کرنے
اور کچھ حصہ کا کاروبار میں استعمال کرونگا۔ میں نے یہ قرض لینے
سے پہلے بنک کے سارے واجب ادا کر دیئے تھے۔ نئے قرضہ واپس
کرنے کی مدت چھ ماہ ہے لیکن میں ساری رقم چار ماہ میں
واپس کر دوں گا۔ پچھلے سال میں نے پوری ادائیگی کے بعد اپنے
اخراجات میں کمی کی تھی اور اپنے قرضے میں سود کو بھی معاف کروایا
تھا اور اپنے پاس اچھے پیسے بھی جمع کر لئے۔ میں ادھار لینے
اس دفعہ دوسرے بنک کے پاس بھی جاؤنگا۔

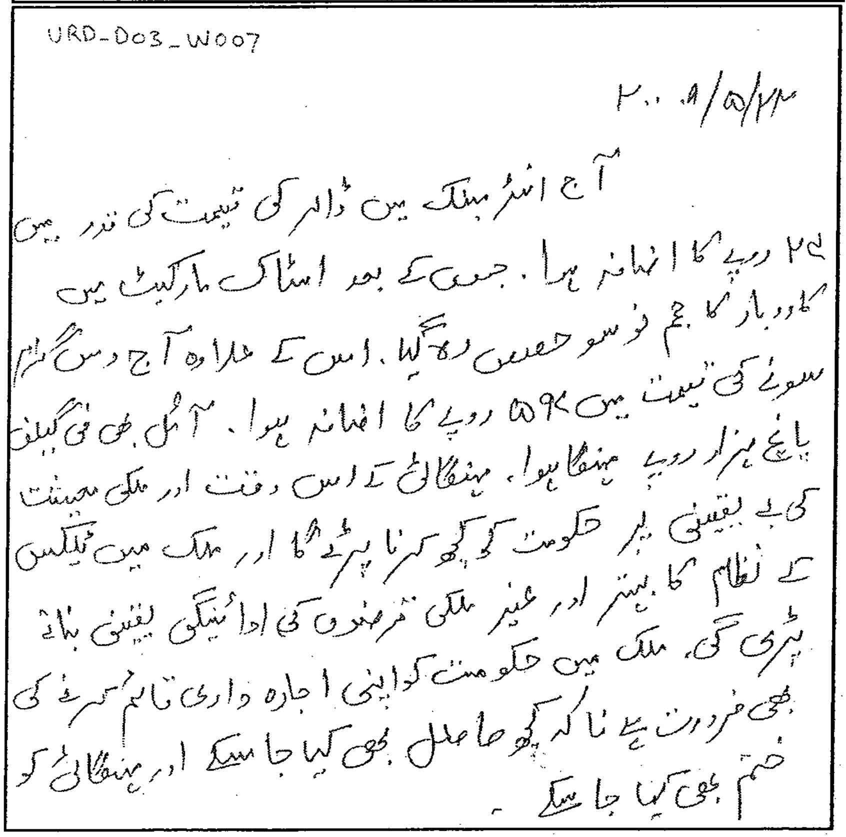

URD_D01_W007

آج مورخہ ۱۷.۱۰.۲۰۰۸ کو ۳۳ ملی میٹر بارش ہوئی اور گرد
کی شدت میں کمی ہوئی اور درجہ حرارت ۲۳ سینٹی گریڈ ہو گیا
میرا قد ۱۹۰ سینٹی میٹر لمبا ہے۔ ہفتے کی فہرست میں ایک
درجن انڈے بھی شامل ہیں اور دودھ کا ڈبہ بھی۔
دوا کی مقدار نے میزان میں کچھ ردوبدل بھی کیا ہے۔ چار دن بعد
میں تحفہ کی حوالگی واپس بناؤں گا۔
آٹھ روز کے بعد بھی شکلیں میں کمی نہیں ہوئی۔
اس کی قیمت دو ہزار تین سو روپے ہے۔

Appendix C: (a) Samples of First Two Handwritten Text Pages


URD_D02_W007

میں نے اس سال کافی مقدار میں کپاس فروخت کی جس کا وزن تقریباً ۲۳۷
ٹن تھا۔ اپنے استعمال کے لئے میں نے کپاس کا کچھ مقدار ذخیرہ بھی کر لیا۔ میں نے
اس سال اپنے شہر میں ایک کمرشل پلاٹ نمبر ۲۳ بھی خریدا۔ یہ پلاٹ
میرے دفتر سے ۷ کلومیٹر کے فاصلے پر واقع ہے۔ میں پلاٹ کی قیمت
کی ادائیگی ایک مہینے میں ادا کی اور زمین کا انتقال میرے نام آٹھ
دن میں ہوا۔ میں نے زمین کے انتقال پر حکومت کو محصول بھی
ادا کیا۔ میں نے اس پلاٹ پر ایک عمارت بنانی ہے۔ جس کی دیوار کی
لمبائی ۶۱ میٹر اور چوڑائی ۶۹ انچ ہے۔ زمین کا کچھ ٹکڑا میں باغ
بنانے کے لیے استعمال کروں گا۔ پلاٹ پر میری لاگت ۲۲۵۶ ہزار
روپے آئی ہے۔


URD_D03_W007

۲۰۰۹/۵/۲۳

آج انٹر بینک میں ڈالر کی قیمت کی قدر میں
۲۷ روپے کا اضافہ ہوا۔ جس کے بعد اسٹاک مارکیٹ میں
کاروبار کا حجم نو سو حصص رہ گیا۔ اس کے علاوہ آج دس گرام
سونے کی قیمت میں ۵۹۲ روپے کا اضافہ ہوا۔ آٹا بھی فی کلو
پانچ ہزار روپے مہنگا ہوا۔ مہنگائی کے اس وقت اور ملکی معیشت
کی بے یقینی پر حکومت کو کچھ کرنا پڑے گا اور ملک میں ٹیکس
کے نظام کا بہتر اور غیر ملکی قرضوں کی ادائیگی یقینی بنانی
پڑے گی۔ ملک میں حکومت کو اپنی اجارہ داری قائم کرنے کی
بھی ضرورت ہے تاکہ کچھ حاصل بھی کیا جا سکے اور مہنگائی کو
ختم بھی کیا جا سکے۔

Appendix C: (b) Samples of Last Two Handwritten Text Pages